# Analisis Kinerja Protokol *Routing Destination Sequence Distance Vector* (DSDV) dan *Optimized Link State Routing* (OLSR) Berdasarkan Mobilitas *Gauss-Markov* Pada *Mobile Ad-hoc Network* (MANET)

**SKRIPSI**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Aditya Prayudhi
NIM: 145150200111092

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

# PENGESAHAN

Analisis Kinerja Protokol *Routing Destination Sequence Distance Vector* (DSDV) dan *Optimized Link State Routing* (OLSR) Berdasarkan Mobilitas *Gauss-Markov* Pada *Mobile Ad-hoc Network* (MANET)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Aditya Prayudhi
NIM: 145150200111092

Skripsi ini telah diuji dan dinyatakan lulus pada
26 Desember 2018
Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Rakhmadhany Primananda, S.T., M.Kom. | Reza Andria Siregar, S.T., M.Kom. |
| NIK: 2016098604061001 | NIP: 19790621 200604 1 003 |

Mengetahui,
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

# PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 26 Desember 2018


Aditya Prayudhi

NIM: 145150200111092

# KATA PENGANTAR

Puji dan syukur penulis panjatkan atas kehadirat Allah SWT karena rahmat dan karunia – Nya, penulis dapat menyelesaikan skripsi yang digunakan sebagai syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika pada Fakultas Ilmu Komputer Universitas Brawijaya.

Untuk kesempatan ini, penulis juga menyampaikan banyak terima kasih kepada pihak-pihak yang telah membantu penulis selama mengerjakan skripsi ini, diantaranya:

1. Allah SWT yang telah memberi kemudahan dalam proses pengerjaan skripsi ini.
2. Kepada kedua orang tua penulis beserta keluarga besar yang selalu memberikan saran, pengertian, motivasi, dan doa.
3. Bapak Rakhmadhany Primananda, S.T., M.Kom., selaku dosen pembimbing I yang telah memberikan banyak waktu kepada penulis selama proses pengerjaan skripsi, baik dalam masukan, saran, motivasi, hingga ilmu yang tentunya sangat bermanfaat bagi penulis.
4. Bapak Reza Andria Siregar S.T., M.Kom., selaku dosen pembimbing II yang telah memberikan banyak waktu kepada penulis selama proses pengerjaan skripsi, baik dalam masukan, saran, motivasi, hingga ilmu yang tentunya sangat bermanfaat bagi penulis.
5. Seluruh Dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberikan ilmu kepada penulis selama masa perkuliahan berlangsung.
6. Teman-teman dekat penulis yang selalu mengingatkan dan memberikan dorongan kepada penulis ketika penulis mengerjakan skripsi ini.

Penulis menyadari bahwa skripsi ini belum dapat dikatakan sempurna. Namun, penulis berharap agar skripsi ini dapat bermanfaat bagi pihak yang membutuhkan.

Malang, 26 Desember 2018

Penulis

Aditya.prayudhi@gmail.com

# ABSTRAK

Teknologi *wi-fi* saat ini sudah sangat berkembang karena memungkinkan berbagai perangkat untuk terhubung ke internet tanpa harus menggunakan kabel sebagai media perantaranya. Namun teknologi tersebut sangat bergantung kepada infrastruktur agar dapat berfungsi. Hal inilah yang kemudian memicu munculnya teknologi *Mobile Ad-Hoc Network* (MANET). Teknologi MANET merupakan teknologi jaringan yang dapat berjalan tanpa adanya infrastruktur terpusat. Dibutuhkan adanya suatu protokol agar perangkat dapat berkomunikasi satu sama lain. Protokol DSDV dan protokol OLSR merupakan contoh protokol yang dapat digunakan pada jaringan MANET dan keduanya bersifat *proactive*. Untuk dapat mengetahui protokol mana yang lebih baik dalam kondisi jaringan tertentu, dibutuhkan adanya simulasi. Simulasi yang banyak digunakan oleh para peneliti adalah dengan menggunakan bantuan *software* karena terkendala biaya yang harus dikeluarkan jika harus melakukan simulasi secara riil. Untuk dapat melakukan simulasi pada lingkungan MANET, dibutuhkan sebuah model atau pola pergerakan yang dapat merepresentasikan setiap perangkat atau *node* dengan akurat. Salah satu contoh model atau pola pergerakan adalah *gauss-markov*. Pada model pergerakan *gauss-markov*, awalnya *node* akan berjalan dengan kecepatan dan arah tertentu, lalu setelah ada interval waktu akan dilakukan penghitungan tujuan dan kecepatan yang selanjutnya berdasarkan tujuan dan kecepatan *node* pada saat itu. Penelitian ini menguji kinerja protokol DSDV dan protokol OLSR dengan menggunakan pergerakan *gauss-markov*. Hasil dari pengujian yang dilakukan, protokol OLSR secara umum memiliki kinerja yang lebih baik. Namun kelebihan dari protokol DSDV ialah nilai *end-to-end delay* yang lebih rendah pada saat pengujian dengan jumlah *node* yang sedikit dan kecepatan *node* yang rendah.

Kata kunci: *Mobile Ad-Hoc Network*, DSDV, OLSR, *Gauss-Markov*

# ABSTRACT

*Wireless technology is well developed because it connects mobile devices to the internet without using any cables attached to the mobile devices.But wireless technolgy, such as wifi is very dependent on the infrastructure. This then triggered the emergence of Mobile Ad-Hoc Network (MANET) technology. MANET technology is a network technology that can run without centralized infrastructure. MANET also needs a protocol so that every devices can communicate with each other. DSDV protocol and OLSR protocol are the protocols commonly used in MANET and both are proactive protocols. Simulation is needed to determine which protocol is better in a certain network condition. Researchers mainly used software simulator to help them simulate MANET because real life simulation costs too high. To be able to simulate the MANET environment, it needs a mobilty model that can accurately represent each device or node. One example of a mobility model is gauss-markov. In the gauss-markov mobility model, initially the node will run with a certain speed and direction, then after some time interval the calculation of destination and speed will be carried out based on the destination and speed of the node at that time. This research examined the performance of the DSDV protocol and the OLSR protocol using the gauss-markov mobility model. The results of the tests conducted, the OLSR protocol generally has better performance. But the advantages of the DSDV protocol are lower end-to-end delay values when testing with fewer nodes and lower node speeds.*

*Keywords: Mobile Ad-Hoc Network*, DSDV, OLSR, *Gauss-Markov*

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB 1 PENDAHULUAN

## 1.1 Latar belakang

Teknologi *wi-fi* saat ini sudah sangat berkembang karena memungkinkan berbagai perangkat untuk terhubung ke internet tanpa harus menggunakan kabel sebagai media perantaranya. Untuk dapat menggunakan *wi-fi*, dibutuhkan setidaknya satu infrastruktur pusat yang disebut dengan *router*. Namun adanya kerusakan pada infrastruktur akan mengakibatkan hubungan antar perangkat dalam jaringan tersebut terputus dan tidak dapat melakukan komunikasi. Hal inilah yang kemudian mempelopori munculnya MANET.

*Mobile Ad-Hoc Network* (MANET) merupakan teknologi jaringan yang mampu berjalan tanpa adanya infrastruktur jaringan komunikasi dan internet seperti *Base Transceiver Station* (BTS), *router*, maupun *access point*. MANET terdiri dari sekumpulan *mobile node* yang saling berkomunikasi satu sama lain secara *multi-hop* (Fatkhurrozi, 2018). Setiap perangkat *mobile* pada MANET dapat bergerak bebas kemanapun sehingga terputusnya komunikasi antar *node* sering terjadi (Raja & Baboo, 2014).

Salah satu protokol yang digunakan pada MANET adalah *Destination Sequence Distance Vector* atau biasa disebut dengan DSDV. Protokol DSDV menggunakan *sequence number* untuk menghindari terjadinya *looping* yang sering terjadi pada MANET. Protokol DSDV merupakan protokol *routing* yang bersifat *proactive* (Narra et al., 2011). Protokol yang bersifat *proactive* lainnya ialah *Optimized Link State Routing* atau biasa disebut dengan OLSR. Protokol ini memanfaatkan *multipoint relay* agar dapat mengurangi jumlah paket *broadcast* yang dikirimkan (Gowrishankar, 2010).

Terdapat dua cara untuk dapat melakukan simulasi pada MANET yakni dengan bantuan software dan percobaan secara riil (*testbed*). Namun mayoritas peneliti lebih memilih simulasi dengan menggunakan bantuan *software* karena terkendala biaya yang harus dikeluarkan jika harus menggunakan percobaan secara riil (*testbed*). Oleh karena itu, simulasi menggunakan *software* merupakan alternatif yang layak dan banyak digunakan oleh para peneliti (Hogie, Bouvry & Guinand, 2006). Untuk dapat melakukan simulasi pada lingkungan MANET, dibutuhkan sebuah model atau pola pergerakan yang dapat merepresentasikan setiap *node* dengan akurat. Harus terdapat perubahan kecepatan dan arah gerak dari *node* agar dapat merepresentasikan *node* dengan baik. Salah satu model atau pola pergerakan adalah *gauss-markov*. Model pergerakan ini awalnya diciptakan untuk dapat melakukan simulasi *Personal Communication Service* (PCS), namun model pergerakan ini juga dapat digunakan untuk melakukan simulasi protokol *ad hoc*. Pada model pergerakan *gauss-markov*, awalnya *node* akan berjalan dengan kecepatan dan arah tertentu, lalu setelah ada interval waktu akan dilakukan penghitungan yang membuat kecepatan dan arah gerak dari *node* berubah. Kecepatan dan arah gerak *node* saat ini mempengaruhi penghitungan kecepatan dan arah gerak *node* yang selanjutnya (Camp, Boleng & Davies, 2002).

Pada penelititan dengan judul "Analisis Kinerja Protokol Routing AD HOC ON-DEMAND DISTANCE VECTOR (AODV) Dan DESTINATION SEQUENCE DISTANCE VECTOR (DSDV) Pada Mobile AD HOC NETWORK (MANET)" yang dilakukan oleh Tanudjaya Tri Ratna Sari pada tahun 2016, membandingkan kinerja dari salah satu protokol *reactive* yakni AODV dengan salah satu protokol *proactive* yakni DSDV dengan menggunakan simulator NS-2. Pola pergerakan yang digunakan pada penelitian tersebut adalah *random waypoint*. Pada penelitian tersebut dapat dilihat dampak dari jumlah *node* dan luas area simulasi terhadap nilai *end-to-end delay*, *packet delivery ratio* (PDR), dan *routing overhead*.

Berdasarkan latar belakang tersebut, maka penulis melakukan penelitian untuk menganalisis kinerja protokol *Destination Sequence Distance Vector* (DSDV) dan *Optimized Link State Routing* (OLSR) dengan pola pergerakan *gauss-markov*. Pengukuran kinerja akan dilihat berdasarkan beberapa parameter seperti *packet delivery ratio*, *end-to-end delay*, dan *routing overhead* dengan variasi jumlah *node*, kecepatan maksimal, dan ukuran paket yang dikirimkan. Penulis memilih kedua protokol di atas karena keduanya memiliki kelebihan masing-masing. Sedangkan pergerakan *gauss-markov* dipilih karena pada penelitian yang berjudul "A Survey of Mobility Models for Ad Hoc Network Research" yang dilakukan oleh Tracy Camp (2002) menyatakan bahwa pergerakan *gauss-markov* mendekati pergerakan *mobile node* yang ada di dunia nyata dibandingkan dengan pergerakan *random* yang lain. Pergerakan ini juga menghindari pergerakan *node* yang berbelok secara tajam pada saat simulasi dijalankan.

## 1.2 Rumusan masalah

Berdasarkan uraian latar belakang di atas, maka dapat dirumuskan permasalahan pada penelitian ini yaitu sebagai berikut:

1. Bagaimana implementasi protokol DSDV dan protokol OLSR dengan pola pergerakan *gauss-markov* pada NS-3?
2. Bagaimana hasil simulasi protokol DSDV dan protokol OLSR dengan pola pergerakan *gauss-markov* pada NS-3?
3. Protokol manakah yang menunjukkan kinerja lebih baik pada saat pengujian?

## 1.3 Tujuan

Tujuan yang ingin dicapai pada penelitian ini adalah:

1. Dapat melakukan implementasi protokol DSDV dan protokol OLSR dengan pola pergerakan *gauss-markov* pada NS-3.
2. Mengetahui hasil simulasi dari protokol DSDV dan protokol OLSR dengan pola pergerakan *gauss-markov* pada NS-3.
3. Mengetahui protokol mana yang menunjukkan kinerja lebih baik saat pengujian.

## 1.4 Manfaat

Penelitian ini diharapkan dapat memberikan beberapa manfaat, diantaranya:

1. Penelitian ini dapat dijadikan pelajaran dalam mengimplementasikan teori yang sudah didapatkan.
2. Penelitian ini dapat dijadikan acuan untuk menentukan protokol yang tepat pada jaringan MANET dengan karakteristik tertentu.

## 1.5 Batasan masalah

Batasan masalah dari penelitian ini adalah:

1. Pengujian dilakukan dengan menggunakan Network Simulator 3.
2. Luas area simulasi sebesar 400x400 meter.
3. Nilai *alpha* pada *gauss-markov* sebesar 0.8.
4. Simulasi dilakukan selama 300 detik.
5. Pengukuran kinerja dari tiap protokol dimulai setelah detik ke 60.
6. Terdapat 10 *node* sumber dan 10 *node* tujuan pada tiap pengujian.
7. Parameter yang menjadi tolak ukur kinerja tiap protokol dalam penelitian ini adalah *packet delivery ratio, end-to-end delay,* dan *routing overhead*.

## 1.6 Sistematika pembahasan

Sistematika pembahasan merupakan penjabaran deskriptif mengenai hal-hal yang akan dibahas dalam penelitian ini, penelitian ini disusun sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan, serta manfaat dari penelitian.

**BAB II LANDASAN KEPUSTAKAAN**

Bab ini berisi tinjauan pustaka dari penelitian yang telah dilakukan sebelumnya serta diuraikan mengenai beberapa dasar teori yang digunakan dalam proses penelitian ini.

**BAB III METODOLOGI**

Bab ini menjelaskan mengenai langkah-langkah yang dilakukan dalam melaksanakan penelitian ini.

**BAB IV PERANCANGAN DAN IMPLEMENTASI**

Bab ini menjelaskan mengenai proses perancangan yang dilakukan dalam penelitian ini. Pada bab ini juga dilakukan tahap implementasi berdasarkan perancangan yang telah dibuat.

**BAB VI PENGUJIAN DAN ANALISIS**

Pada bab ini penulis akan menjelaskan mengenai proses pengujian dan menguraikan hasil analisis dari pengujian yang telah dilakukan pada penelitian ini.

**BAB VI PENUTUP**

Pada bab ini penulis menuliskan beberapa kesimpulan yang diperoleh dari penelitian yang dilakukan serta saran untuk pengembangan penelitian yang selanjutnya.

# BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi tinjauan pustaka dari penelitian sebelumnya serta uraian dan pembahasan mengenai teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian.

## 2.1 Tinjauan Pustaka

**Tabel 2.1 Tinjauan Pustaka**

| No | Nama Penulis, Tahun, Judul Penelitian | Persamaan | Perbedaan | |
|---|---|---|---|---|
| | | | Penelitian Terdahulu | Rencana Penelitian |
| 1 | Tanudjaya Tri Ratna Sari (2016) Analisis Kinerja Protokol Routing AD HOC ON-DEMAND DISTANCE VECTOR (AODV) Dan DESTINATION SEQUENCE DISTANCE VECTOR (DSDV) Pada Mobile AD HOC NETWORK (MANET) | Melakukan analisis kinerja protokol *routing* menggunakan NS-3 | Pergerakan menggunakan *random waypoint* | Pergerakan menggunakan *gauss-markov* |
| 2 | Fatkhurrozi (2018) Analisis Perbandingan Kinerja Protokol AOMDV, DSDV, Dan ZRP Sebagai Protokol Routing Pada Mobile Ad-Hoc Network (MANET) | Melakukan analisis kinerja protokol *routing* | Skenario pengujian menggunakan variasi luas area | Skenario pengujian menggunakan variasi jumlah *node*, variasi kecepatan maksimal *node*, dan variasi ukuran paket |

## 2.2 Dasar Teori

### 2.2.1 *Mobile Ad-Hoc Network*

Saat orang terkoneksi dengan jaringan dengan perangkatnya, orang tersebut masih ingin tetap berada pada jaringan tersebut walaupun dia harus bergerak. Namun jika dia berpindah dari suatu jaringan ke jaringan yang lain maka alamat dari perangkat yang digunakannya pun akan berubah dan paket yang harusnya

dikirimkan pada alamat tersebut tidak akan sampai. *Router* tidak menyimpan alamat dari alamat baru yang digunakan oleh perangkat yang tidak terhubung lagi. Beberapa solusi seperti *Mobile* IP*,* DHCP*,* dan *Cellular Networks* dapat digunakan untuk dapat mendukung pergerakan seperti pada masalah tersebut. Namun semua teknologi yang telah disebutkan tadi masih bergantung pada infrastruktur. Atas dasar inilah dipilih *mobile ad-hoc network* sebagai solusinya (Gupta, 2016).

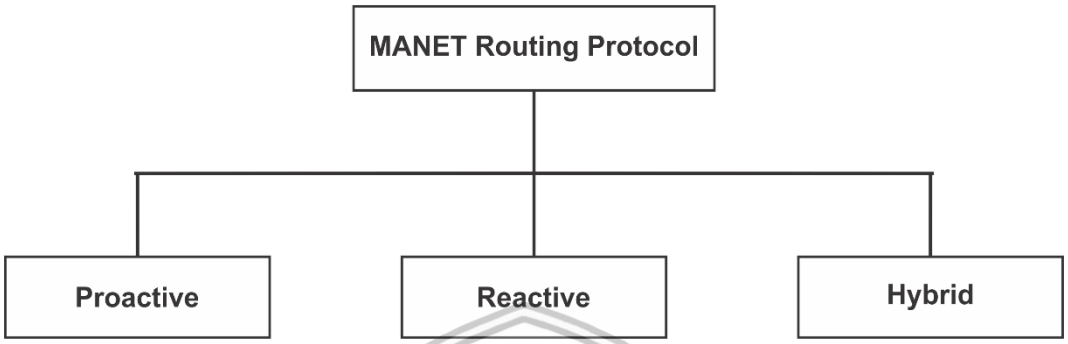
*Mobile Ad-Hoc Network* (MANET) merupakan teknologi jaringan yang mampu berjalan tanpa adanya infrastruktur jaringan komunikasi dan internet seperti BTS, router, maupun access point. MANET terdiri dari sekumpulan *mobile node* yang saling berkomunikasi satu sama lain secara *multi-hop* (Fatkhurrozi, 2018). Setiap *node* berhubungan satu sama lain melalui jaringan nirkabel atau *wireless*. Perangkat dalam MANET dapat bebas bergerak kemana saja, sehingga topologi jaringannya dapat berubah secara tidak menentu dan dinamis. Tujuan dari MANET adalah mendukung jaringan *mobile* dengan menambahkan funsionalitas *routing* ke dalam perangkat tersebut. MANET telah melahirkan banyak aplikasi seperti *Tactical Network, Wireless Sensor Network, Data Networks,* dan *Device Networks* (Raja & Baboo, 2014).

Terdapat beberapa karakteristik pada MANET yang membedakannya dengan infrastruktur jaringan lainnya yaitu topologi jaringan yang bersifat dinamis, batasan daya, batasan *bandwidth*, keamanan, dan juga batasan memori penyimpanan. Karakteristik pertama ialah *node* pada MANET yang selalu bergerak dan menyebabkan terjadinya perubahan topologi. Oleh karena itu, topologi jaringan hanya akan berlaku untuk waktu yang sangat singkat. Hal ini membuat protokol yang telah digunakan pada jaringan kabel tidak akan cocok digunakan pada MANET. Jenis *node* pada MANET yang mayoritas merupakan perangkat bergerak tentunya bergantung pada daya baterai yang digunakan. Oleh karena itu, protokol yang digunakan pada MANET dirancang dengan memperhatikan manajemen daya yang akan digunakan untuk dapat menjalankannya dengan baik (Gupta, 2016).

Seperti yang telah dikatakan bahwa mayoritas *node* pada MANET merupakan perangkat bergerak, maka tentu saja *node* tersebut menggunakan koneksi *wireless*. Hal ini tentu memiliki perbedaan *bandwidth* dengan perangkat yang terkoneksi menggunakan kabel. Karakteristik selanjutnya ialah keamanan pada MANET. Seharusnya tidak seorang pun yang dapat melihat data personal 1 melacaknya selama proses transmisi data. Mekanisme enkripsi yang tepat harus digunakan untuk melindungi privasi penggunanya. Dan yang terakhir adalah mengenai masalah batasan memori penyimpanan pada MANET. *Node mobile* tentu saja memiliki kapabilitas untuk melakukan perhitungan dan penyimpanan informasi yang lebih rendah dibandingkan dengan *node* yang statis (Gupta, 2016).

Dalam melakukan pengiriman data dari *source* menuju *destination*, diperlukan adanya sebuah protokol *routing*. Dibutuhkan beberapa komunikasi dengan *node* terdekat untuk dapat mencapai *destination*. Protokol *routing* pada MANET umumnya dibagi menjadi tiga jenis, yaitu (Fatkhurrozi, 2018):

1. Protokol *routing* reaktif.
2. Protokol *routing* proaktif.
3. Protokol *routing hybrid.*




**Gambar 2.1 Jenis Protokol Pada MANET**
**Sumber: (Fatkhurrozi, 2018)**

Protokol *routing* yang bersifat *proactive* menggunakan *routing table* untuk menyimpan informasi mengenai rute menuju semua *node*. Pada *routing table* terdapat informasi berupa *node* mana yang harus dilewati jika akan menuju ke suatu *node* walaupun pada saat itu rute tersebut akan digunakan atau tidak. Tabel pada tiap *node* harus selalu melakukan *update* karena topologi pada MANET yang selalu berubah. Contoh protokol yang bersifat *proactive* adalah *Destination Sequence Distance Vector* dan *Optimized Link State Routing Protocol*.

Protokol *routing* yang bersifat *reactive* dirancang untuk mengatasi masalah *overhead* yang umumnya terdapat pada protokol *proactive* karena topologi jaringan pada MANET yang dinamis. Protokol yang bersifat *reactive* melakukan pencarian rute hanya jika rute tersebut dibutuhkan untuk melakukan komunikasi dengan *node* yang lain. Proses *maintain* rute juga hanya dilakukan pada rute yang saat itu sedang digunakan sehingga mengurangi beban pada jaringan. Contoh protokol yang bersifat *reactive* adalah *Ad-hoc On Demand Distance Vector* dan *Dynamic Source Routing*.

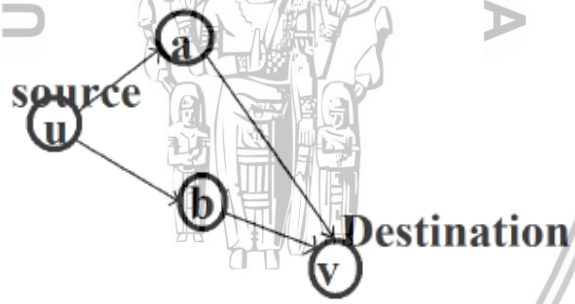
Protokol yang bersifat *hybrid* muncul karena adanya kekurangan pada protokol *proactive* dan protokol *reactive*. Banyak pula yang membutuhkan sebuah protokol yang menggabungkan karakteristik yang bagus dari *proactive* dan *reactive*. Salah satu protokol yang bersifat *hybrid* adalah *Zone Routing Protocol* (Gupta, 2016).

### 2.2.2 DSDV

*Destination Sequence Distance Vector* (DSDV) menggunakan algoritme Bellman-Ford untuk melakukan penentuan jalurnya. Salah satu variabel yang digunakan adalah *hop count*, atau banyaknya *hop* yang harus ditempuh oleh paket agar dapat mencapai tujuannya. DSDV adalah protokol *routing* yang bersifat *proactive* dan juga *table-driven* sehingga protokol ini melakukan *maintain routing table* dengan semua *node* yang ada dalam ruang lingkupnya, tidak hanya *node* tetangganya saja. Perubahan pada *routing table* terjadi secara periodik atau jika

ada *trigger* untuk melakukan *update*. Setiap *node* akan melakukan *broadcast* tentang *routing table* yang dimilikinya ke seluruh *node* yang berada dalam area cakupannya. *Node* tetangga yang menerima pesan tersebut akan melakukan *update* juga dan seterusnya hingga seluruh *node* telah menerima informasi perubahan *routing table* dari *node* yang pertama melakukan *update*. Mekanisme seperti ini yang digunakan oleh protokol DSDV untuk *maintain* hubungan terhadap *node* lainnya.

Masalah seperti adanya *looping* bisa muncul saat melakukan *update* pada *routing table*. Jika terjadi *looping,* maka *routing table* pada *node* tersebut menjadi tidak *valid.* Lebih parahnya lagi, informasi mengenai *routing table* tersebut akan dikirimkan ke *node* lainnya dan akhirnya membuat banyak paket tidak dapat dikirimkan dengan benar. Untuk menghindari masalah tersebut, pada protokol DSDV setiap *update* yang dilakukan oleh tiap *node* akan diberikan *tag* berupa *sequence number*. Pada setiap adanya perubahan, normalnya tiap *sequence number* merupakan angka genap, oleh karena itu *sequence number* dari *node* tersebut akan ditambah dua setiap melakukan *update*. Jika sebuah *node* ingin melakukan *update* mengenai adanya rute yang sudah *expired*, maka *sequence number* dari *node* tersebut akan ditambah satu agar menjadi angka ganjil. *Node* yang menerima pesan *update* ini akan melihat adanya *sequence number* yang ganjil dan akan membuang jalur tersebut dari *routing table*-nya (Narra et al., 2011).

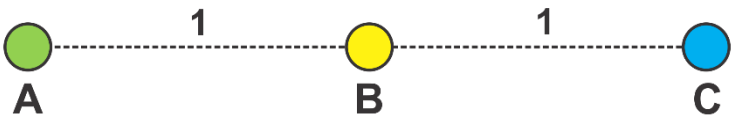



**Gambar 2.2 Protokol DSDV**
**Sumber: (R & Kumar, 2014)**

Pada Gambar 2.2, *node* u sebagai *node* asal mengirimkan informasi *routing table* pada *node* a dan *node* b. Kedua *node* tersebut memiliki informasi tentang *node* v sebagai *node* tujuan. *Node* a dan *node* b kemudian mengirimkan informasi mengenai *routing table* mereka pada *node* u. Algoritme Bellman-Ford untuk menentukan jalur mana yang merupakan jalur terbaik dan jalur tersebut akan digunakan untuk mengirimkan paket dari *node* u ke *node* v. Variabel lain seperti jarak atau *cost* akan digunakan untuk melihat *node* mana yang seharusnya dilewati agar dapat mengirimkan data ke *node* tujuan (R & Kumar, 2014).

Protokol DSDV bersifat *proactive* yang berarti setiap *node* harus selalu mengetahui rute yang harus dilewati untuk menuju ke *node* lainnya. Pada protokol DSDV, *node* bertukar pesan *hello* ke tetangganya untuk melakukan *advertise*. Lalu *node* tetangganya juga akan membalas pesan *hello* tersebut, sehingga *node* tersebut saling mengetahui satu sama lain. Hal tersebut terjadi pada seluruh *node*

yang terdapat pada jaringan sehingga pada akhirnya semua *node* mengenal tetangganya masing-masing. Setelah itu setiap *node* akan melakukan *update* pada *routing table* untuk dapat mengetahui jalur yang harus ditempuh. Gambar 2.3 merupakan ilustrasi dari *routing table* antar *node*. *Node* A, B, dan C pada gambar tersebut sudah mengenal tetangga masing-masing dan terus melakukan *update* untuk dapat mengetahui jalur menuju *node* lainnya (Daas et al., 2015).




| Dest. | Next | Metric | ... |
|---|---|---|---|
| A | A | 0 | |
| B | B | 1 | |
| C | B | 2 | |

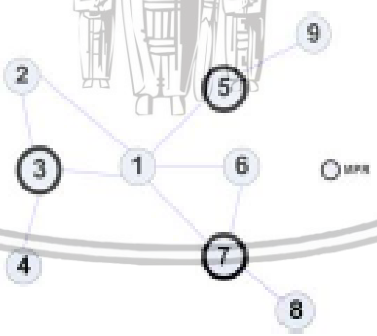
| Dest. | Next | Metric | ... |
|---|---|---|---|
| A | A | 1 | |
| B | B | 0 | |
| C | C | 1 | |

| Dest. | Next | Metric | ... |
|---|---|---|---|
| A | B | 2 | |
| B | B | 1 | |
| C | C | 0 | |

**Gambar 2.3 *Routing Table* Protokol DSDV**
**Sumber: (Daas et al., 2015)**

### 2.2.3 OLSR

*Optimized Link State Routing Protocol* (OLSR) merupakan protokol *routing* yang bersifat *proactive*. Protokol ini memanfaatkan *multipoint relay* (MPR) untuk mengoptimalkan kinerjanya. Dengan menggunakan MPR, protokol OLSR dapat mengurangi jumlah paket *broadcast* yang dikirimkan. Setiap *node* akan melakukan pemilihan *node* MPR dalam setiap kumpulan tetangga *1-hop*. Dengan cara demikian, jumlah paket yang harus ditransmisikan dalam jaringan akan berkurang (Gowrishankar, 2010).



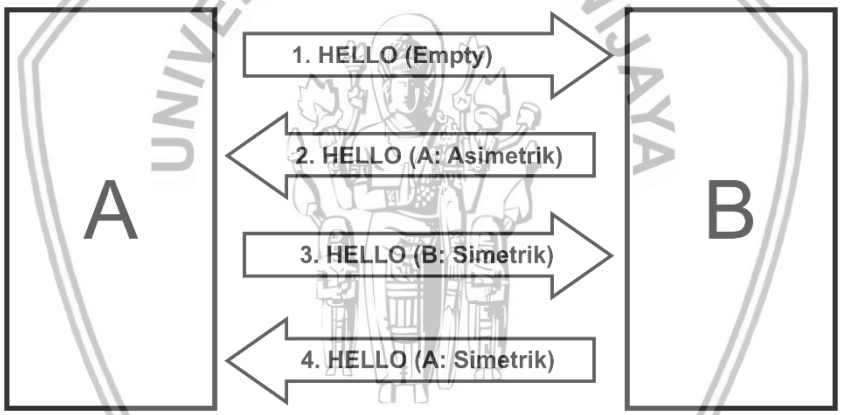

**Gambar 2.4 Protokol OLSR**
**Sumber: (Gowrishankar, 2010)**

Gagasan tentang MPR muncul untuk mengurangi *flooding* pada jaringan dengan cara mengurangi pengiriman paket-paket yang bersifat *reduntdant*. Setiap *node* memilih sekumpulan *node* dalam tetangga *1-hop* yang disebut dengan MPR. *Node* MPR akan menyimpan informasi mengenai setiap *node* yang memilihnya sebagai MPR. Sebuah *node* mendapatkan informasi tersebut melalui pesan HELLO yang dikirimkan secara periodik oleh MPR. Kumpulan *node* MPR yang telah dipilih harus dapat melingkupi seluruh *node* pada tetangga 2-hop seperti yang ditunjukkan pada gambar 2.4 di atas agar dapat mengirimkan informasi ke seluruh

*node* pada jaringan. Semakin sedikit jumlah MPR yang ada dalam suatu jaringan, maka semakin sedikit pula *overhead* yang dihasilkan oleh protokol tersebut. Keberhasilan protokol OLSR sangat bergantung pada mekanisme pemilihan MPR yang tepat, karena MPR merupakan *node* antara dalam sebuah jalur yang akan meneruskan proses pengiriman data.

Dalam protokol OLSR, penentuan jalur yang digunakan untuk mengirim paket dari *node* sumber ke *node* penerima mempertimbangkan jumlah *hop* yang harus ditempuh. Ketika sebuah paket melewati sebuah *node*, maka dapat dikatakan paket tersebut telah menempuh satu *hop*. Makin sedikit jumlah *hop* yang harus ditempuh maka makin besar jalur tersebut dipilih menjadi jalur terbaik untuk menuju ke *node* tujuan (Clausen & Jacquet, 2003).

Optimasi pada protokol OLSR didapatkan dengan dua cara. Pertama dengan mengurangi jumlah paket *broadcast* yang harus dikirimkan lewat jaringan tersebut karena hanya beberapa *node* yang disebut MPR yang akan melakukan *broadcast*. Kedua dengan berkurangnya ukuran dari paket *control* yang dikirimkan karena informasi mengenai anggota dalam kelompok tetangganya saja yang dikirim, bukan informasi mengenai seluruh *node* pada jaringan (Gowrishankar, 2010).

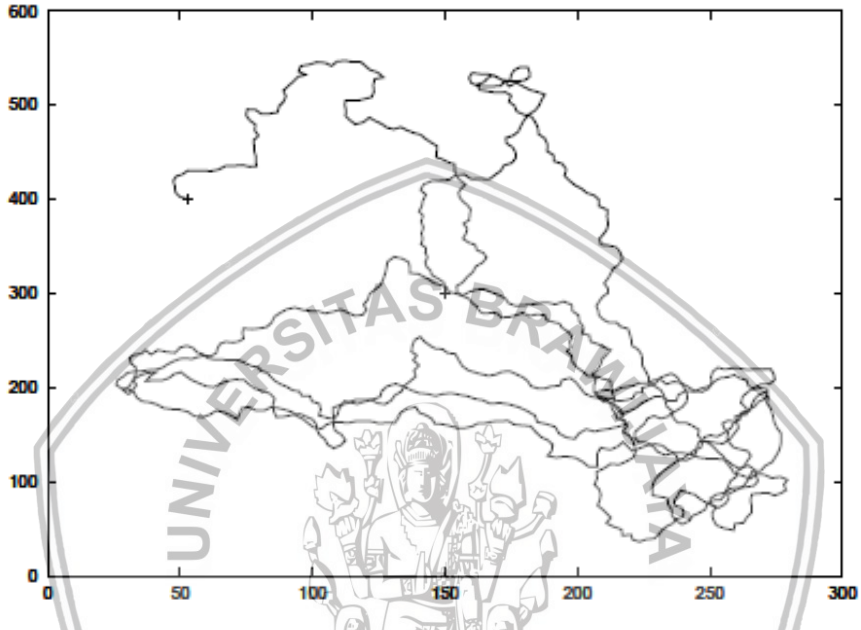



**Gambar 2.5 Mekanisme Penelusuran Tetangga Protokol OLSR**
**Sumber: (http://www.olsr.org)**

Gambar 2.5 merupakan mekanisme penelusuran tetangga pada protokol OLSR. Proses penelusuran ini dilakukan oleh seluruh *node* yang terdapat pada jaringan. Pada gambar tersebut, *node* A mengirimkan pesan *hello* tanpa status adanya status *link.* Kemudian *node* B akan mendaftakan *node* A sebagai tetangga asimetrik terlebih dahulu dan mengirimkan pesan *hello* kembali pada *node* A. Selanjutnya *node* A akan menerima pesan dari *node* B dimana terdapat alamat dari *node* A itu sendiri sehingga A mendaftarkan *node* B sebagai tetangga simetrik lalu kembali mengirimkan pesan *hello.* Saat *node* B menerima pesan tersebut, maka B akan mengetahui bahwa *node* A merupakan tetangga simetrik dan pada akhirnya mendaftarkan *node* A sebagai tetangga simetrik lalu mengirimkan pesan *hello* kembali ke *node* A.

### 2.2.4 *Gauss-Markov*

Model pergerakan *gauss-markov* awalnya diciptakan untuk dapat melakukan simulasi *Personal Communication Service* (PCS), namun model ini juga dapat digunakan untuk simulasi protokol *ad hoc*. Pada model *gauss-markov*, *node* akan diberikan arah gerak dan kecepatan awal, setelah periode waktu tertentu akan dilakukan perhitungan yang membuat kecepatan dan arah gerak dari *node* berubah.




**Gambar 2.6 Pergerakan *Gauss-Markov***
**Sumber: (Camp, Boleng & Davies, 2002)**

Untuk mencegah sebuah *node* agar tidak berada di tepi area simulasi untuk waktu yang lama, dengan menggunakan model pergerakan *gauss-markov node* tersebut akan merubah arah geraknya. Sebagai contoh, jika sebuah *node* berada di tepi kanan area simulasi untuk waktu tertentu, penetuan tujuan selanjutnya dari *node* tersebut akan menjauhi tepi kanan dari area simulasi. Gambar 2.6 di atas merupakan contoh pergerakan sebuah *node* dengan model pergerakan *gauss-markov*. *Node* tersebut awalnya berada pada titik tengah area simulasi dan simulasi dijalankan selama 1000 detik (Camp, Boleng & Davies, 2002).

$$\boldsymbol{s_n = \alpha s_{n-1} + (1-\alpha)\overline{s} + \sqrt{(1-\alpha^2)}\, s_{Xn-1}} \quad (2.1)$$

$$\boldsymbol{d_n = \alpha d_{n-1} + (1-\alpha)\overline{d} + \sqrt{(1-\alpha^2)}\, d_{Xn-1}} \quad (2.2)$$

Dua rumus di atas adalah rumus yang digunakan dalam model pergerakan *gauss-markov* dalam menentukan pergerakan *node*. Pada rumus tersebut, $s_n$ adalah kecepatan baru sedangkan $d_n$ adalah arah gerak yang baru. Variabel α merupakan parameter yang menentukan keacakan dari pergerakan. Variabel α sendiri dapat bernilai $0 \leq \alpha \leq 1$. Semakin kecil nilai α maka pergerakan *node* semakin acak dan semakin besar nilai α maka pergerakan *node* akan semakin mendekati pergerakan dengan model *random waypoint*. Sedangkan variabel $\overline{s}$ dan

$\overline{d}$ merupakan variabel dengan nilai yang tetap yang mencerminkan rata-rata kecepatan dan arah gerak dari *node*. Dengan menggunakan kecepatan dan arah gerak sebelumnya dari *node* untuk menentukan kecepatan dan arah gerak yang selanjutnya, pergerakan *gauss-markov* dapat menghilangkan masalah seperti *node* yang berhenti secara mendadak maupun *node* yang berbelok dengan tajam.

### 2.2.5 *Network Simulator* 3

*Network Simulator* 3 atau biasa disebut NS-3 merupakan simulator yang bersifat *open-source* yang digunakan untuk tujuan riset dan edukasi. NS-3 mulai digunakan sejak tahun 2006 dan saat ini mendapat lisensi dari GNU GPLv2. NS-3 dapat mendukung simulasi yang menggunakan Wi-Fi, WiMax, LTE dan banyak protokol *routing* yang digunakan pada MANET. Beberapa karakteristik dari NS-3 adalah (Manpreet, 2014):

1. Menggunakan bahasa C++ dan Python.
2. Lebih baik dibandingkan NS-2 dalam hal manajemen memori.
3. Menyediakan dokumentasi yang baik.
4. Mendukung penggunaan pada *virtual machine* yang ringan.
5. NS-3 bukan merupakan versi terbaru dari NS-2, beberapa API pada NS-2 tidak dapat digunakan pada NS-3.
6. Memiliki beberapa fitur yang lebih baik dari NS-2 dan terus melakukan pengembangan.

Selain hal-hal yang sudah disebutkan di atas terdapat juga beberapa batasan pada NS-3, yakni (Manpreet, 2014):

1. Karena menggunakan Python, dukungan untuk visualisasi pun terbatas.
2. Tidak cocok dengan NS-2, sehingga susah untuk berpindah menggunakan NS-3.

### 2.2.6 AWK

Awk adalah bahasa pemrograman yang dapat digunakan untuk melakukan tugas seperti pengambilan data, pemrosesan data, dan pembuatan laporan agar lebih mudah. Awk pada awalnya diperkenalkan pada tahun 1979, dan mulai digunakan pada kalangan luas walaupun yang tidak mempunyai latar belakang mengenai pemrograman sekalipun.

Operasi dasar yang digunakan pada awk adalah memeriksa satu set masukan satu per satu dan melakukan pencarian untuk mendapatkan pola maupun kondisi yang cocok yang telah dimasukkan oleh pengguna. Dapat diberikan sebuah aksi tertentu untuk setiap polanya. Aksi tersebut akan dijalankan pada tiap baris yang cocok dengan pola.

Gambar 2.7 di bawah merupakan pola penulisan *script* awk. Beberapa baris dapat berupa perintah untuk mencetak dan baris lainnya untuk memproses pola yang sederhana. Pola seperti *regular expression* juga bisa digunakan untuk

menyeleksi baris yang diinginkan. Ada dua bagian spesial pada awk, yakni *BEGIN* dan *END* sebagai penanda baris yang dibaca dan akhir dari baris yang sudah diproses (Aho, Kernighan & Weinberger, 1979).

```
Pattern    {action}
Pattern    {action}
...
```

**Gambar 2.7 Penulisan *Script* awk**
**Sumber: (Aho, Kernighan & Weinberger, 1979)**

### 2.2.7 *Packet Delivery Ratio*

*Packet delivery ratio* (PDR) adalah rasio antara jumlah paket data yang diterima oleh tujuan dengan jumlah paket data yang dikirim dari asalnya. Secara matematis, PDR dapat dirumuskan sebagai berikut (Rohal, Dahiya, & Dahiya, 2013):

$$PDR = \frac{S1}{S2} X\ 100\% \quad (2.3)$$

PDR : *packet delivery ratio*
S1 : jumlah total paket data yang diterima
S2 : jumlah total paket data yang dikirimkan

### 2.2.8 *End-to-End Delay*

*End-to-end delay* adalah waktu yang ditempuh oleh suatu paket data untuk dapat mencapai tujuannya. Cara untuk mendapatkan nilai *end-to-end delay* adalah dengan mengurangi waktu pada saat paket data sampai di *node* tujuan dengan waktu pada saat paket data dikirimkan oleh *node* asal (Rohal, Dahiya, & Dahiya 2013).

### 2.2.9 *Routing Overhead*

Sebagian besar protokol *routing* mengirimkan *control information* sebagai salah satu proses agar protokol dapat menjalankan fungsinya. Contohnya adalah mengenai pengaksesan saluran *broadcast* dalam TCP. *Control information* tersebut merupakan *overhead* saat protokol berjalan. Akibatnya, dapat mengurangi *bandwidth* yang seharusnya dipakai untuk pengiriman data. Pada beberapa kasus, *control information* dihsasilkan secara periodik (Shaukat & Syrotiuk, 2013).

*Node* tujuan dapat menerima lebih dari satu *Route Request* (RREQ) dari tetangga *1-hop*, yang berarti *node* pengirim atau *node* asal dapat menerima lebih dari satu *Route Reply* (RREP) dari berbagai *node* yang lain. Jadi banyaknya pengiriman pesan seperti itu menjadi penyebab adanya *control overhead* (Hazra & Setua, 2010).

# BAB 3 METODOLOGI

Penelitian ini merupakan penelitian non-implementatif analitik dan metodologi yang digunakan dalam penelitian ini meliputi beberapa tahapan, yaitu studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis. Berikut diagram metodologi dalam penelitian ini:


**Gambar 3.1 Diagram Metodologi Penelitian**

## 3.1 Studi Literatur

Studi literatur berisi tentang tinjauan pustaka dari penelitian sebelumnya dan dasar teori yang digunakan dalam penelitian ini. Dasar teori didapatkan dari beberapa sumber seperti skripsi, jurnal, dan artikel. Dasar teori yang digunakan dalam penelitian ini adalah dua protokol (DSDV dan OLSR) pada MANET, pola

pergerakan pergerakan *gauss-markov*, AWK sebagai alat bantu untuk mendapatkan hasil simulasi, dan *Quality of Service* (QoS) sebagai parameter hasil diantaranya: *packet delivery ratio*, *end-to-end delay*, dan *routing overhead*.

## 3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk menganalisa semua kebutuhan yang diperlukan dalam membangun simulasi yang akan dilakukan pada penelitian ini, sehingga nantinya dapat diperoleh kebutuhan yang sesuai untuk simulasi. Kebutuhan dibagi menjadi dua yaitu kebutuhan fungsional dan kebutuhan *non*-fungsional.

### 3.2.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan tentang apa proses apa saja yang dilakukan dan hasil apa yang didapatkan. Kebutuhan fungsional dalam penelitian ini yaitu sebagai berikut:

1. Mensimulasikan protokol OLSR dan protokol DSDV pada jaringan MANET dengan pola pergerakan *Gauss-Markov* dengan tiga skenario pengujian.
2. Dari simulasi yang dijalankan dapat menghasilkan data yang berisi kinerja kedua protokol sehingga dapat dilakukan analisis dari data yang didapatkan.

### 3.2.2 Kebutuhan *Non*-Fungsional

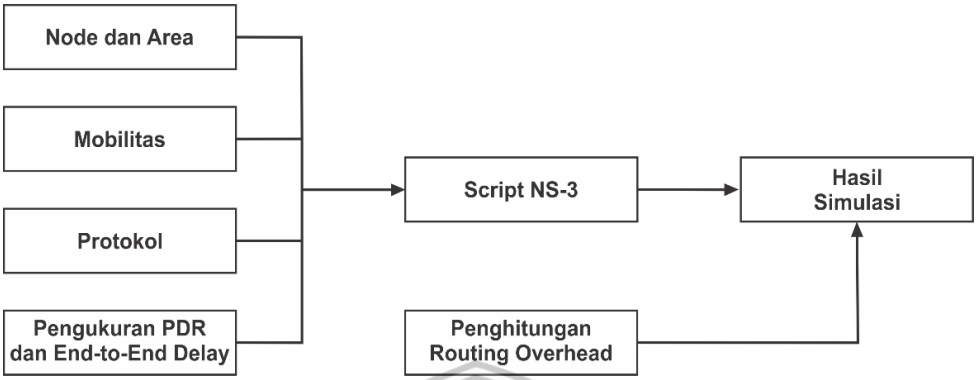
Kebutuhan *non*-fungsional adalah perangkat yang digunakan pada penelitian ini untuk dapat membangun dan menjalankan lingkungan simulasi sesuai dengan skenario pengujian. Kebutuhan *non*-fungsional dalam penelitian ini yaitu sebagai berikut:

1. Perangkat Keras
    - 1 Buah PC dengan spesifikasi :
        - *Processor* Intel Core i7 CPU @2.50GHz
        - RAM 12GB
2. Perangkat Lunak
    - Sistem operasi Windows 10 dan Ubuntu 14
    - Network Simulator 3

## 3.3 Perancangan dan Implementasi

Pada tahap ini akan dijelaskan mengenai skenario yang digunakan dalam penelitian serta proses perancangan simulasi. Gambar 3.2 merupakan skema perancangan yang digunakan dalam penelitian ini. Tahap-tahap yang dilakukan ialah merancang *node* dan area simulasi, merancangan mobilitas *node*, merancang protokol yang digunakan, dan merancang pengukuran parameter berupa *packet delivery ratio* dan *end-to-end delay*. Rancangan tadi akan dijadikan satu dalam sebuah *script* NS-3. Setelah *script* berhasil dijalankan, akan dilakukan

penghitungan *routing overhead* dari hasil simulasi tersebut menggunakan bantuan *script* awk. Setelah tahap perancangan selesai, langkah selanjutnya adalah mengimplementasikan rancangan yang telah dibuat ke dalam NS-3.




**Gambar 3.2 Skema Perancangan Simulasi**

### 3.3.1 Perancangan *Node* dan Area Simulasi

Tahap perancangan yang pertama dilakukan adalah merancang *node* dan area simulasi yang akan digunakan. Jumlah *node* yang akan dibuat disesuaikan dengan jumlah *node* yang dibutuhkan pada simulasi. Tahap ini juga akan mendefinisikan luas area simulasi yang akan digunakan dalam simulasi agar terdapat batas area yang jelas untuk pergerakan dari *node*.

### 3.3.2 Perancangan Mobilitas *Node*

Setelah selesai merancang *node* dan area simulasi, yang selanjutnya dilakukan adalah merancang mobilitas yang digunakan oleh *node*. Mobilitas *node* akan menentukan bagaimana perilaku pergerakan dari tiap *node*. Terdapat beberapa variabel dalam mobilitas, salah satunya ialah kecepatan *node*. Penentuan kecepatan *node* akan disesuaikan dengan kebutuhan simulasi.

### 3.3.3 Perancangan Parameter Protokol

Tahap selanjutnya ialah merancang parameter protokol yang akan digunakan. Sebuah protokol diperlukan agar *node* dapat saling mengirim pesan. Pada tahap ini akan didefinisikan protokol yang dibutuhkan dalam simulasi. Terdapat dua protokol yang akan digunakan pada penelitian ini, yakni protokol *Destination Sequence Distance Vector* (DSDV) dan protokol *Optimized Link State Routing* (OLSR). Selain itu juga akan didefinisikan kebutuhan lain yang mendukung berjalannya simulasi.

### 3.3.4 Perancangan Paramater Uji

Parameter uji yang digunakan sebagai tolak ukur kinerja protokol DSDV dan OLSR dalam penelitian ini adalah *packet delivery ratio*, *end-to-end delay*, dan *routing overhead*. Pada tahap ini, akan dilakukan perancangan tiga parameter uji tersebut.

## 3.4 Pengujian dan Analisis

Pada tahap ini akan dilakukan pengujian dari hasil implementasi yang telah dibuat dan menampilkan hasil pengujian berdasarkan skenario yang telah dirancang. Setelah menampilkan data hasil pengujian akan dilakukan analisis berdasarkan hasil yang didapatkan dari pengujian tersebut.

## 3.5 Penutup

Pada bagian Penutup akan berisi kesimpulan dan saran dari penulis. Kesimpulan didapatkan dari hasil pengujian dan analisis yang telah dilakukan dan dari kesimpulan yang telah didapat akan muncul saran yang berisi usulan dari penulis untuk membantu penelitian yang akan datang mengenai *Mobile Ad Hoc Network*.

# BAB 4 PERANCANGAN DAN IMPLEMENTASI

Bab perancangan dan implementasi menjelaskan tentang proses perancangan mulai dari perancangan *node* dan area simulasi, mobilitas *node*, protokol yang digunakan, hingga perancangan penghitungan parameter pengujian. Pada bab ini juga dilakukan implementasi dari perancangan yang telah dibuat.

Terdapat beberapa hal yang harus dijabarkan sebelum dapat menjalankan simulasi yaitu protokol *routing*, jumlah *node*, mobilitas *node*, kecepatan minimum dan maksimum, luas area simulasi, dan ukuran paket yang dikirimkan. Pada penelitian ini terdapat tiga skenario pengujian yang akan digunakan, yakni skenario dengan variasi jumlah *node*, skenario dengan variasi kecepatan maksimal *node*, dan skenario dengan variasi ukuran paket.

**Tabel 4.1 Spesifikasi Skenario Variasi Jumlah *Node***

| PARAMETER | NILAI |
|---|---|
| Protokol *Routing* | DSDV dan OLSR |
| Jumlah *Node* | 20, 40, 60, dan 80 *node* |
| Mobilitas | *Gauss-Markov* |
| Kecepatan Minimum | 10 m/s |
| Kecepatan Maksimum | 30 m/s |
| Luas Area Simulasi | 400x400 |
| Waktu Simulasi | 300 detik |
| Ukuran Paket | 256 *bytes* |

Berikut merupakan spesifikasi skenario variasi jumlah *node* yang akan digunakan pada penelitian ini:

1. Protokol *routing* yang digunakan adalah *Destination Sequence Distance Vector* (DSDV) dan *Optimized Link State Routing* (OLSR).
2. Variasi jumlah *node* pada simulasi adalah 20, 40, 60, dan 80 *node*. Kepadatan *node* pada jaringan yang paling rendah pada penelitian ini direpresentasikan oleh simulasi dengan 20 *node* dan kepadatan *node* pada jaringan yang paling tinggi direpresentasikan oleh simulasi dengan 80 *node*.
3. Mobilitas yang digunakan selama simulasi adalah *gauss markov*.
4. Kecepatan minimum sebuah *node* adalah 10 m/s.
5. Kecepatan maksimum sebuah *node* adalah 30 m/s.
6. Luas area simulasi adalah 400x400.

7. Waktu simulasi adalah 300 detik.
8. Ukuran paket yang dikirim adalah 256 *bytes*.

**Tabel 4.2 Spesifikasi Skenario Variasi Kecepatan *Node***

| PARAMETER | NILAI |
|---|---|
| Protokol *Routing* | DSDV dan OLSR |
| Jumlah *Node* | 40 *node* |
| Mobilitas | *Gauss-Markov* |
| Kecepatan Minimum | 5 m/s |
| Kecepatan Maksimum | 15, 20, 25, dan 30 m/s |
| Luas Area Simulasi | 400x400 |
| Waktu Simulasi | 300 detik |
| Ukuran Paket | 256 *bytes* |

Berikut merupakan spesifikasi sistem skenario variasi kecepatan *node* yang akan digunakan pada penelitian ini:

1. Protokol *routing* yang digunakan adalah *Destination Sequence Distance Vector* (DSDV) dan *Optimized Link State Routing* (OLSR).
2. Jumlah *node* yang digunakan adalah 40 *node*.
3. Mobilitas yang digunakan selama simulasi adalah *gauss markov*.
4. Kecepatan minimum sebuah *node* adalah 5 m/s.
5. Variasi kecepatan maksimum sebuah *node* adalah 15, 20, 25, dan 30 m/s. *Node* dengan tingkat mobilitas yang paling rendah pada penelitian ini direpresentasikan pada simulasi dengan kecepatan maksimum sebesar 15 m/s dan *node* dengan tingkat mobilitas yang paling tinggi direpresentasikan pada simulasi dengan kecepatan maksimum sebesar 30 m/s.
6. Luas area simulasi adalah 400x400.
7. Waktu simulasi adalah 300 detik.
8. Ukuran paket yang dikirim adalah 256 *bytes*.

**Tabel 4.3 Spesifikasi Skenario Variasi Ukuran Paket**

| PARAMETER | NILAI |
|---|---|
| Protokol *Routing* | DSDV dan OLSR |
| Jumlah *Node* | 40 *node* |
| Mobilitas | *Gauss-Markov* |

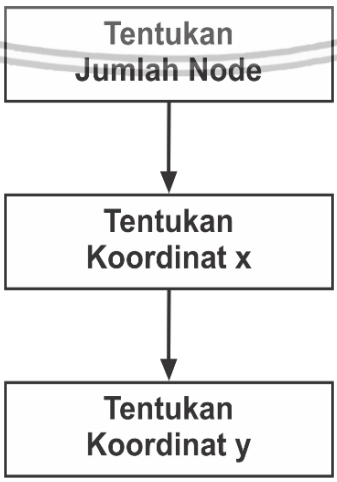
| | |
|---|---|
| Kecepatan Minimum | 10 m/s |
| Kecepatan Maksimum | 30 m/s |
| Luas Area Simulasi | 400x400 |
| Waktu Simulasi | 300 detik |
| Ukuran Paket | 256, 1024, 4096, 8192, dan 12288 *bytes* |

Berikut merupakan spesifikasi sistem skenario variasi ukuran paket yang akan digunakan pada penelitian ini:

1. Protokol *routing* yang digunakan adalah *Destination Sequence Distance Vector* (DSDV) dan *Optimized Link State Routing* (OLSR).
2. Jumlah *node* yang digunakan adalah 40 *node*.
3. Mobilitas yang digunakan selama simulasi adalah *gauss markov*.
4. Kecepatan minimum sebuah *node* adalah 10 m/s.
5. Variasi kecepatan maksimum sebuah *node* adalah 30 m/s.
6. Luas area simulasi adalah 400x400.
7. Waktu simulasi adalah 300 detik.
8. Ukuran paket yang dikirim adalah 256, 1024, 4096, 8192, dan 12288 *bytes*. Paket dengan ukuran terkecil pada penelitian ini direpresentasikan pada simulasi dengan ukuran paket sebesar 256 *bytes* dan paket dengan ukuran terbesar direpresentasikan dengan ukuran 12288 *bytes*.

## 4.1 Perancangan

### 4.1.2 Perancangan *Node* dan Area Simulasi



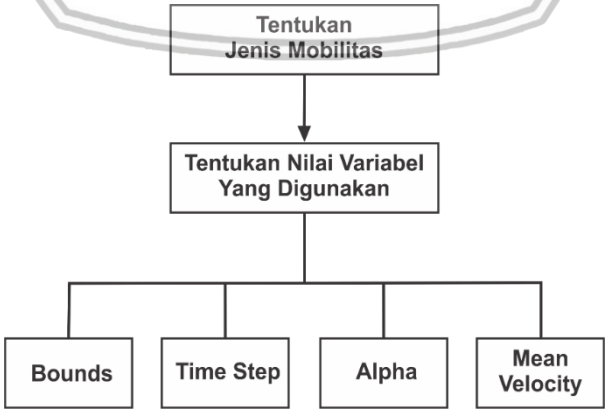

**Gambar 4.1 Skema Perancangan *Node* dan Area Simulasi**

Langkah pertama yang dilakukan adalah merancang *node* dan area simulasi. Dalam merancang *node* dan area simulasi terdapat tahap-tahap perancangan yang harus dilakukan. Tahap-tahap tersebut adalah menentukan jumlah *node*, menentukan koordinat x, dan menentukan koordinat y. Tahapan tersebut dapat dilihat pada Gambar 4.1 di atas.

Pada tahap ini akan ditentukan jumlah *node* dan ukuran dari area simulasi yang digunakan. Jumlah *node* yang dibutuhkan dalam penelitian ini bervariasi, yakni sebanyak 20, 40, 60, dan 80 *node*. Sedangkan area simulasi yang dibutuhkan dalam penelitian ini sebesar 400x400 meter. Koordinat pada sumbu x yang digunakan dimulai dari 0 hingga 400 dan koordinat pada sumbu y yang digunakan dimulai dari 0 hingga 400. Gambar 4.2 di bawah merupakan rancangan posisi atau letak 20 *node* pada area simulasi dengan ukuran 400x400. Pada gambar tersebut, *node* digambarkan berupa lingkaran dengan angka di tengahnya sebagai penanda tiap *node*nya. Karena posisi *node* ditentukan secara acak, maka ilustrasi di bawah tidak dapat menggambarkan posisi *node* yang sebenarnya.


**Gambar 4.2 Perancangan *Node* dan Area Simulasi**

### 4.1.3 Perancangan Mobilitas *Node*

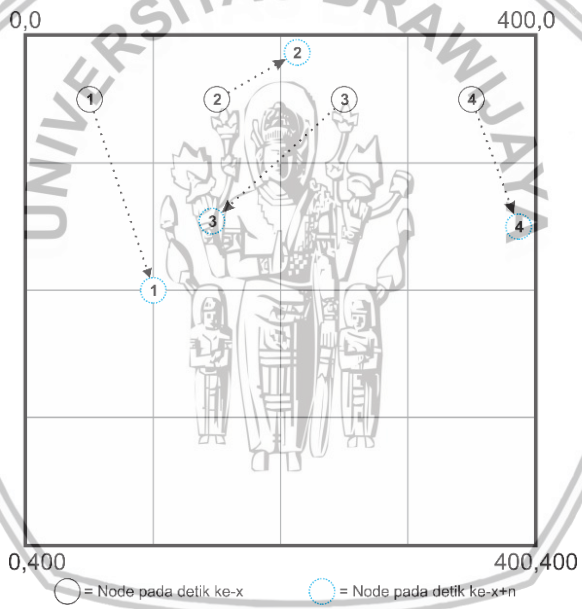



**Gambar 4.3 Skema Perancangan Mobilitas *Node***

Setelah *node* dan area simulasi selesai dirancang, langkah selanjutnya ialah merancang mobilitas *node*. Dalam merancang mobilitas *node* terdapat tahap-tahap perancangan yang harus dilakukan. Tahap-tahap tersebut adalah

menentukan jenis mobilitas dan menentukan nilai dari variabel yang akan digunakan. Variabel tersebut berupa *bounds* sebagai luas area simulasi, *time step* sebagai waktu jeda, *alpha* sebagai faktor yang menentukan gerakan *node*, dan *mean velocity* sebagai penentu kecepatan *node*. Tahapan tersebut dapat dilihat pada Gambar 4.3 di atas.

Pada penelitian ini, jenis mobilitas yang digunakan adalah *gauss-markov.* Dalam mobilitas *gauss-markov* terdapat beberapa variabel seperti yang telah dijelaskan di atas. Pada penelitian ini, nilai dari variabel *bounds* akan mengikuti ukuran dari area simulasi yakni 400x400, variabel *time step* akan diberikan nilai sebesar 0.5 detik, variabel *alpha* akan diberikan nilai sebesar 0.8, dan nilai variabel *mean velocity* akan bervariasi dan disesuaikan dengan skenario yang digunakan. Gambar 4.4 menunjukkan perancangan mobilitas dan kecepatan *node*. Pada saat simulasi berlangsung, diharapkan setiap *node*nya dapat bergerak sesuai dengan jenis mobilitas dan kecepatan yang sudah ditentukan. Pada gambar tersebut terdapat 4 *node* sebagai contoh dan tiap *node*nya akan bergerak dengan arah dan kecepatan yang berbeda.

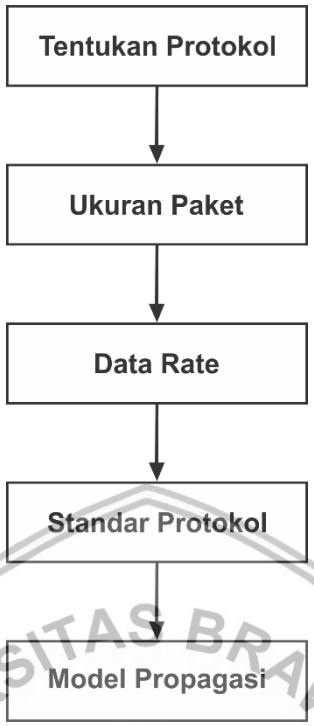



**Gambar 4.4 Perancangan Mobilitas *Node***

### 4.1.4 Perancangan Parameter Protokol

Langkah selanjutnya ialah merancang parameter protokol yang akan digunakan dalam simulasi. Dalam merancang parameter protokol terdapat tahap-tahap perancangan yang harus dilakukan. Tahap-tahap tersebut adalah menentukan protokol, menentukan ukuran paket, menentukan *data rate*, menentukan standard protokol yang digunakan, dan menentukan model propagasi. Tahapan tersebut dapat dilihat pada Gambar 4.5 di bawah.

Pada penelitian ini protokol yang digunakan adalah DSDV dan OLSR, ukuran paket yang dikirim bervariasi dan disesuaikan dengan skenario yang digunakan, *Data rate* yang digunakan sebesar 2048 bps, standar protokol yang digunakan adalah 802.11b dan model propagasi yang digunakan adalah *FriisPropagationLoss*.




**Gambar 4.5 Skema Perancangan Parameter Protokol**

### 4.1.5 Perancangan Pengukuran Parameter Uji

Pada bagian ini akan dijelaskan tentang perancangan pengukuran parameter pengujian berupa *packet delivery ratio*, *end-to-end delay,* dan *routing overhead*. *Packet delivery ratio* dan *end-to-end delay* didapatkan dengan menambahkan beberapa baris kode dalam *script* NS-3 yang akan dibuat. Sedangakan *routing overhead* didapatkan dengan bantuan awk.

Seperti yang telah dijelaskan pada bab 2, *packet delivery ratio* adalah rasio antara jumlah paket yang diterima oleh *node* tujuan dengan jumlah paket yang dikirimkan oleh *node* sumber. Untuk mendapatkan nilai *packet delivery ratio* pada simulasi ini akan digunakan bantuan *flowmonitor* yang dapat menampilkan jumlah paket yang diterima dan dikirimkan. Lalu akan dilakukan penghitungan rasio dan hasilnya akan ditampilkan dalam bentuk persentase.

Seperti yang telah dijelaskan pada bab 2, *end-to-end delay* adalah waktu yang dibutuhkan oleh paket untuk mencapai tujuannya. Parameter ini didapatkan dengan melihat selisih antara waktu saat paket dikirimkan dengan waktu saat paket tersebut sampai pada tujuan. Untuk mendapatkan nilai *end-to-end delay* pada simulasi ini akan digunakan bantuan *flowmonitor*. Hasil penghitungan akan ditampilkan dalam satuan *milisecond*.

Pada penelitian ini, yang dimaksud dengan *routing overhead* adalah jumlah paket *routing* yang dikirimkan oleh semua *node* dalam jaringan selama simulasi berlangsung. Seperti yang sudah dijelaskan pada bab 2, protokol *routing* akan mengirimkan *control information* agar protokol dapat menjalankan fungsinya. Untuk mendapatkan nilai *routing overhead*, pada simulasi ini digunakan *script* awk

untuk menghitung jumlah total paket *routing* pada *file trace*. *File trace* merupakan *file* yang akan dihasilkan setelah *script* simulasi selesai dijalankan.

## 4.2 Implementasi

### 4.2.1 Implementasi *Node* dan Area Simulasi

Bagian ini akan membahas tentang implementasi dari perancangan *node* dan area simulasi yang telah dibahas di atas. Langkah awal yang dilakukan adalah menentukan terlebih dahulu jumlah *node* yang akan dibuat. Setelah itu *node* akan ditempatkan pada posisinya masing-masing dalam area simulasi. Digunakan bantuan fungsi fungsi *RandomRectanglePositionAllocator* dan juga fungsi *UniformRandomVariable* untuk menentukan koordinat yang nantinya menjadi posisi awal *node*.

**Tabel 4.4 Potongan Kode Implementasi *Node* dan Area Simulasi**

```
1  int nWifis = 20;
2  NodeContainer adhocNodes;
3  adhocNodes.Create (nWifis);
4
5  ObjectFactory pos;
6  pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
   pos.Set ("X", StringValue
7  ("ns3::UniformRandomVariable[Min=0.0|Max=400.0]"))
   pos.Set ("Y", StringValue
8  ("ns3::UniformRandomVariable[Min=0.0|Max=400.0]"));
```

Tabel 4.4 di atas merupakan potongan kode untuk menentukan banyaknya *node* yang akan digunakan pada saat simulasi dan menempatkan setiap *node*nya pada area simulasi. Nilai dari Variabel nWifis dapat diubah sesuai dengan spesifikasi sistem yang telah dibuat sebelumnya. Contoh kode yang terdapat pada tabel di atas akan membuat *node* sebanyak 20 *node*. Lalu menempatkan tiap *node* sesuai dengan koordinat yang didapat secara acak pada area simulasi. Luas area simulasi yang akan dibuat bergantung kepada nilai minimum dan maksimum dari koordinat *node*.

### 4.2.2 Implementasi Mobilitas *Node*

Setelah *node* sudah ditempatkan pada area simulasi, tahap selanjutnya ialah mengatur mobilitas *node*. Setelah menentukan mobiltas apa yang digunakan dalam simulasi, akan ditentukan nilai dari batas pergerakan *node* (*bounds*), waktu jeda yang digunakan (*time step*), variabel *alpha*, dan juga kecepatan minimum dan maksimum dari *node* (*mean velocity*).

**Tabel 4.5 Potongan Kode Implementasi Mobilitas *Node***

```
1  MobilityHelper mobilityAdhoc;
2  mobilityAdhoc.SetMobilityModel
   ("ns3::GaussMarkovMobilityModel",
3                "Bounds", BoxValue (Box (0, 400, 0, 400, 0,
   0)),
4                "TimeStep", TimeValue (Seconds (0.5)),
                 "Alpha", DoubleValue (0.8),
```

```
                "MeanVelocity", StringValue
("ns3::UniformRandomVariable[Min=10|Max=20]"),
                "MeanDirection", StringValue
("ns3::UniformRandomVariable[Min=0|Max=6.283185307]"),
                "MeanPitch", StringValue
("ns3::UniformRandomVariable[Min=0.05|Max=0.05]"),
                "NormalVelocity", StringValue
("ns3::NormalRandomVariable[Mean=0.0|Variance=0.0|Bound=0.0]
"),
                "NormalDirection", StringValue
("ns3::NormalRandomVariable[Mean=0.0|Variance=0.2|Bound=0.4]
"),
                "NormalPitch", StringValue
("ns3::NormalRandomVariable[Mean=0.0|Variance=0.02|Bound=0.0
4]")
              );

```

Tabel 4.5 di atas merupakan potongan kode untuk menentukan mobilitas apa yang akan digunakan dalam simulasi sekaligus menentukan kecepatan minimum dan maksimum yang dapat dimiliki oleh *node*. Dalam hal ini, mobilitas yang digunakan adalah *gauss-markov*. Pada potongan kode di atas, ditentukan nilai dari parameter yang digunakan pada mobilitas *gauss*-markov, seperti ***bounds*** yang nilainya sama dengan luas area simulasi, waktu jeda (*timestep*) yang bernilai 0.5, variabel *alpha* dengan nilai 0.8, dan *mean velocity* yang digunakan untuk menentukan kecepatan minimum dan maksimum dari *node*. Pada contoh di atas, kecapatan minimumnya sebesar 10 m/s dan kecepatan maksimumnya sebesar 20 m/s. Nilai dari parameter lain yang telah disebutkan di atas dapat diubah dan disesuaikan dengan spesifikasi sistem yang telah dibuat sebelumnya

### 4.2.3 Implementasi Parameter Protokol

Setelah mobilitas selesai diimplementasikan, tahap selanjutnya ialah mengimplementasikan rancangan parameter protokol yang digunakan dalam simulasi agar *node* dapat berinteraksi satu sama lain. Beberapa hal yang menjadi acuan dalam mengimplementasikan parameter protokol adalah jenis protokol yang akan digunakan, ukruan paket, besarnya ukuran data yang dapat ditransmisikan (*data rate*), standar jaringan, dan model propagasi yang digunakan.

**Tabel 4.6 Potongan Kode Implementasi Parameter Protokol**

```
#include "ns3/olsr-module.h"
#include "ns3/dsdv-module.h"

RoutingExperiment::RoutingExperiment ()
 : port (9),
   bytesTotal (0),
   packetsReceived (0),
   m_traceMobility (false),
   m_protocol (2) // 1=OLSR 2=DSDV

OlsrHelper olsr;
DsdvHelper dsdv;

Ipv4ListRoutingHelper list;
```

```
15  InternetStackHelper internet;
16
17  switch (m_protocol)
18    {
19      case 1:
20        list.Add (olsr, 100);
21        m_protocolName = "OLSR";
22        break;
23      case 2:
24        list.Add (dsdv, 100);
25        m_protocolName = "DSDV";
26        break;
27      default:
29        NS_FATAL_ERROR ("No such protocol:" << m_protocol);
30    }
31
32  std::string rate ("2048bps");
33  std::string phyMode ("DsssRate11Mbps");
34  Config::SetDefault
    ("ns3::OnOffApplication::PacketSize",StringValue ("8192"));
35  Config::SetDefault ("ns3::OnOffApplication::DataRate",
    StringValue (rate));
36
37  WifiHelper wifi;
38  wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
39
40  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
41  YansWifiChannelHelper wifiChannel;
42  wifiChannel.SetPropagationDelay
    ("ns3::ConstantSpeedPropagationDelayModel");
43  wifiChannel.AddPropagationLoss
    ("ns3::FriisPropagationLossModel");
44  wifiPhy.SetChannel (wifiChannel.Create ());
```

Tabel 4.6 di atas merupakan potongan kode yang berfungsi untuk mengimplementasikan perancangan parameter protokol yang sudah dibuat sebelumnya. Langkah pertama adalah menentukan protokol yang digunakan, pada contoh di atas protokol yang digunakan adalah DSDV. Pemilihan protokol menggunakan bantuan fungsi *switch case*. Setelah protokol dipilih, langkah selanjutnya adalah menentukan ukuran paket yang dikirim dan besar *data rate* yang digunakan. Pada contoh di atas ukuran paketnya sebesar 8192 *bytes* dengan *data rate* sebesar 2048 *bit per second*. Kemudian yang terakhir adalah menentukan standar protokol dan model propagasi yang digunakan. Dapat dilihat pada contoh di atas, simulasi yang akan dibuat menggunakan standar 802.11b dengan model propagasi *FriisPropagationLossModel*.

### 4.2.4 Implementasi Pengukuran Parameter Uji

Setelah langkah-langkah di atas berhasil diimplementasikan, *script* NS-3 sudah dapat menjalankan simulasi namun diperlukan adanya tambahan kode agar dapat mengukur parameter seperti *packet delivery ratio* dan *end-to-end delay*. Digunakan bantuan *flowmonitor* untuk dapat menampilkan hasil dari kedua parameter tersebut.

**Tabel 4.7 Potongan Kode Implementasi PDR dan *End-to-End Delay***

```
1   for  (std::map<FlowId,
    FlowMonitor::FlowStats>::const_iterator iter = stats.begin
    (); iter != stats.end (); ++iter)
2   {
3     NS_LOG_UNCOND(" ");
4     NS_LOG_UNCOND("Flow ID: " << iter->first);
5     NS_LOG_UNCOND("Tx Packets = " << iter >second.txPackets);
6     NS_LOG_UNCOND("Rx Packets = " << iter->second.rxPackets);
7     NS_LOG_UNCOND("Average Delay = " << (iter-
    >second.delaySum.GetSeconds()/iter->second.rxPackets) *
    1000 << " ms");
8
9     totalRcvdPackets += iter->second.rxPackets;
10    totalSentPackets += iter->second.txPackets;
11    totalDelaySum += iter->second.delaySum.GetSeconds();
12    totalTime += iter->second.timeLastRxPacket.GetSeconds() -
    iter->second.timeFirstTxPacket.GetSeconds();
13  }
14
15  std::cout<< " " << std::endl;
16  std::cout<< "####   SIMULATION RESULT  ####" <<std::endl;
17  std::cout<< "Title = " << tr_name << std::endl;
18  std::cout<< "Sent Packets = " << totalSentPackets <<
19  std::endl;
20  std::cout<< "Received Packets = " << totalRcvdPackets <<
21  std::endl;
22  std::cout<< "Lost Packets = " << totalSentPackets -
23  totalRcvdPackets << std::endl;
24  std::cout<< "PDR = " << (totalRcvdPackets/totalSentPackets)
    * 100 << "%" << std::endl;
25  std::cout<< "Average Delay = " << (totalDelaySum /
    totalRcvdPackets) * 1000 << " ms" << std::endl;
```

Tabel 4.7 di atas merupakan potongan kode untuk mendapatkan nilai *packet delivery ratio* dan rata-rata *end-to-end delay* pada yang akan ditampilkan pada akhir simulasi. Untuk mendapatkan nilai *packet delivery ratio* diperlukan informasi tentang jumlah paket yang dikirimkan dan juga jumlah paket yang berhasil diterima. Setelah itu, jumlah paket yang diterima dibagi dengan jumlah paket yang dikirimkan lalu dikalikan dengan 100 untuk mendapatkan nilai *packet delivery ratio*. Nilai *end-to-end delay* didapatkan dari total waktu *end-to-end delay* dibagi dengan banyaknya paket yang berhasil diterima lalu dikalikan dengan 1000 untuk mengubahnya menjadi satuan *milisecond*.

Pada penelitian ini, *routing overhead* didapatkan dengan cara menghitung seluruh paket *routing* yang dikirimkan oleh semua *node* selama simulasi berlangsung. Simulasi yang berhasil dijalankan akan menghasilkan *file trace* yang di dalamnya berisi riwayat pengiriman selama simulasi. Untuk mendapatkan nilai *routing overhead* dibutuhkan *script* awk untuk dapat membantu melakukan penghitungan jumlah paket routing yang dikirim pada *file trace*.

**Tabel 4.8 Potongan Kode Implementasi Penghitungan *Routing Overhead***

```
1   BEGIN {
2         sent=0
```

```
3       }
4
5       {
6           dsdvPacket=$49
7           #Menghitung jumlah paket routing yang dikirim/di-
    forward
8           if ( dsdvPacket == "ns3::dsdv::DsdvHeader")
9           {sent++;}
10      }
11
12      END {
13          printf("Routing Overhead DSDV = %.0f\n", sent);
14  }
```

Tabel 4.8 di atas merupakan potongan kode dari *script* awk yang digunakan untuk mengetahui berapa jumlah paket *routing* yang dikirimkan selama simulasi dijalankan. *Script* di atas dijalankan setelah simulasi selesai dan akan membaca file trace yang dihasilkan dari simulasi. *Script* tersebut hanya akan menghitung jumlah paket *routing* yang dikirimkan dari dimulainya simulasi hingga selesai. Contoh di atas merupakan *script* untuk menghitung *routing overhead* pada protokol DSDV.

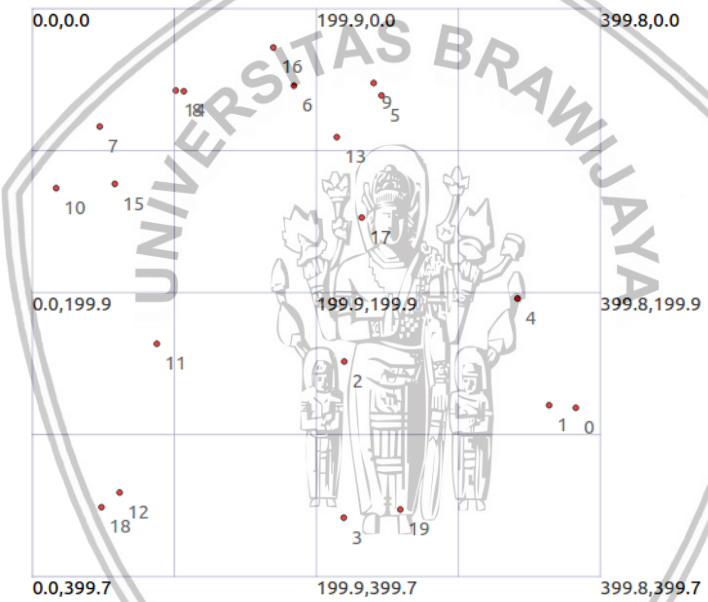
# BAB 5 PENGUJIAN DAN ANALISIS

Pada bab ini akan dilakukan pengujian dengan menjalankan beberapa skenario yang telah dirancang. Setelah semua pengujian selesai, akan dilakukan analisis berdasarkan hasil yang didapatkan dari pengujian tersebut.

## 5.1 Pengujian

### 5.1.1 Pengujian *Node* dan Area Simulasi

Pada bagian perancangan *node* dan area simulasi yang telah dilakukan pada tahap sebelumnya, terdapat 20 *node* pada area simulasi dengan ukuran 400x400. Pada tahap ini akan dilakukan pengujian terhadap perancangan tersebut. Pengujian dilakukan dengan tujuan untuk melihat apakah perancangan *node* dan area simulasi dapat diaplikasikan pada NS-3 atau tidak.

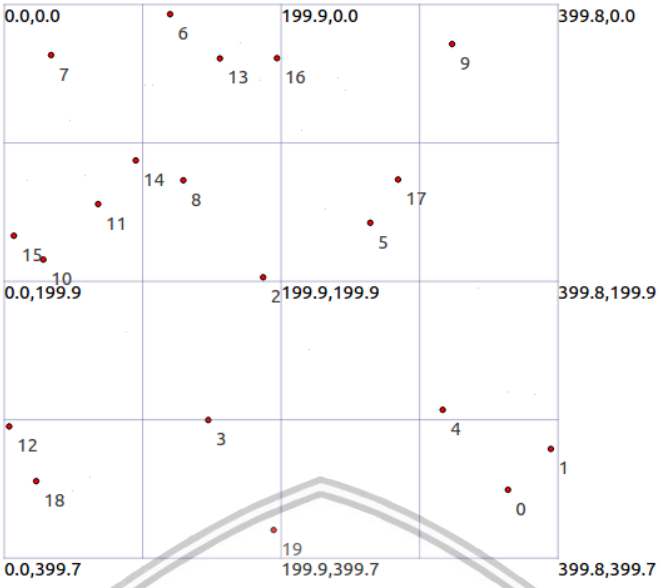



**Gambar 5.1 Pengujian *Node* dan Area Simulasi**

Setelah rancangan selesai dibuat, kode tersebut lalu dijalankan dan akan menghasilkan file baru dengan format xml. File xml tersebut dapat dibuka dengan NetAnim dan memberikan visualisasi dari simulasi yang dibuat. Gambar 5.1 di atas merupakan visualisasi dari pengujian *node* dan area simulasi yang dilakukan menggunakan bantuan NetAnim. Terdapat 20 titik merah yang tersebar pada area simulasi. Setiap titik merah tersebut mewakili satu *node* yang masing masing memiliki identitas berupa angka dari 0 hingga 19.

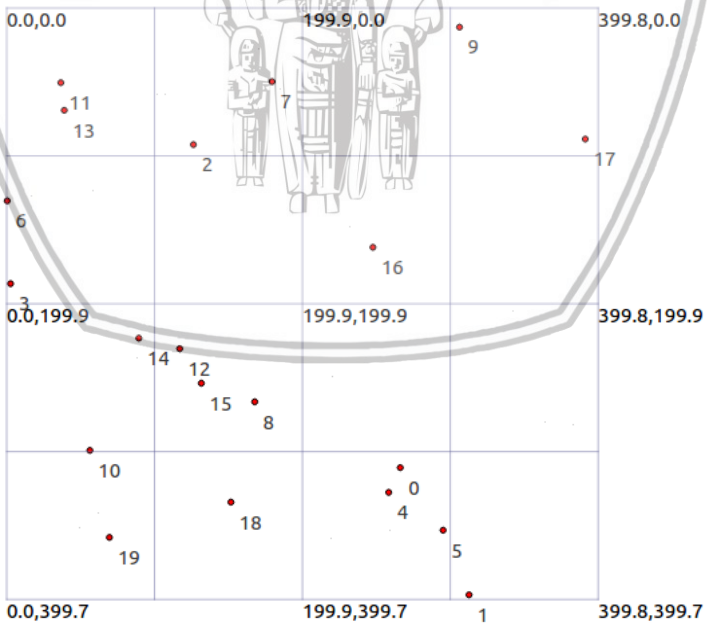
### 5.1.2 Pengujian Mobilitas *Node*

Pada bagian sebelumnya mengenai mobilitas *node* telah ditentukan bahwa mobilitas yang digunakan adalah *gauss-markov*. Pada tahap ini akan dilakukan pengujian dari implementasi yang telah dibuat sebelumnya. Pengujian dilakukan dengan tujuan untuk melihat apakah perancangan mobilitas *node* dapat diaplikasikan pada NS-3 atau tidak.




**Gambar 5.2 Pengujian Mobilitas Pada Detik Ke-5**

Sama seperti pengujian sebelumnya, untuk melihat berhasil atau tidaknya pengujian mobilitas *node*, diperlukan bantuan NetAnim untuk dapat memberikan visualisasi dari simulasi yang dibuat. Posisi awal *node* sebelum simulasi dimulai dapat dilihat pada Gambar 5.1 dan pada Gambar 5.2 menunjukkan adanya perubahan posisi setelah simulasi berjalan selama 5 detik. *Node* terlihat bergerak secara acak dengan kecepatan yang berbeda.




**Gambar 5.3 Pengujian Mobilitas Pada Detik Ke-15**

Gambar 5.3 menunjukkan posisi *node* pada saat simulasi berjalan selama 15 detik. Dibandingkan dengan gambar 5.2 yang menunjukkan posisi *node* saat detik ke-5, perpindahan posisi tiap *node* semakin terlihat jelas pada Gambar 5.3 di atas.

### 5.1.3 Pengujian Parameter Protokol

Pada bagian perancangan parameter protokol telah ditentukan bahwa protokol yang digunakan adalah protokol DSDV dan protokol OLSR. Pada tahap ini akan dilakukan pengujian terhadap perancangan tersebut. Pengujian dilakukan dengan tujuan untuk melihat apakah kedua protokol tersebut dapat diimplementasikan dengan benar pada NS-3 atau tidak.

```
Flow ID : 1
Tx Packets = 239
Rx Packets = 207
Average Delay = 13.6007 ms

Flow ID : 2
Tx Packets = 239
Rx Packets = 190
Average Delay = 7.47282 ms

Flow ID : 3
Tx Packets = 239
Rx Packets = 198
Average Delay = 1.41029 ms

Flow ID : 4
Tx Packets = 239
Rx Packets = 193
Average Delay = 1.66961 ms
```

**Gambar 5.4 Pengujian Parameter Protokol**

Sedikit berbeda dengan pengujian sebelumnya yang menggunakan bantuan NetAnim untuk dapat melihat apakah pengujian tersebut berhasil atau tidak, pengujian ini dilakukan dengan cara menjalankan kode yang telah dibuat dan saat prosesnya selesai, akan ditampilkan informasi seperti jumlah paket yang dikirim, jumlah diterima, dan juga rata-rata *delay* dari tiap *flow*. Jika *node* dalam simulasi berhasil mengirimkan paket berarti implementasi dari protokol sudah berjalan dengan benar. Gambar 5.4 di atas merupakan tampilan yang muncul saat proses selesai. Gambar tersebut merupakan tampilan dari pengujian dengan protokol DSDV.

### 5.1.4 Pengujian Pengukuran Parameter Uji

Pada bagian perancangan parameter pengujian, terdapat tiga parameter yang akan digunakan untuk mengukur kinerja dari protokol DSDV dan protokol OLSR pada mobilitas *gauss-markov*. Parameter yang dimaksudkan adalah *packet delivery ratio, end-to-end delay,* dan *routing overhead*. Pada tahap ini akan dilakukan pengujian terhadap rancangan tersebut. Pengujian dilakukan dengan tujuan untuk melihat apakah setelah simulasi berhasil dijalankan, nilai dari parameter yang disebutkan tadi bisa didapatkan atau tidak.

Untuk mendapatkan nilai *packet delivery ratio* dan *end-to-end delay*, cara yang digunakan adalah dengan menjalankan kode yang telah dibuat dan menunggu hingga prosesnya selesai. Jika berhasil maka akan muncul tampilan seperti pada Gambar 5.5 yang menunjukkan nilai *packet delivery ratio* dan juga *end-to-end*

*delay*. Gambar tersebut merupakan tampilan dari pengujian pada protokol DSDV dengan jumlah *node* 20, kecepatan minimum sebesar 10 m/s dan kecepatan maksimum sebesar 40 m/s.

```
####  SIMULATION RESULT  ####
Title = DSDV_Uji_20Nodes
Sent Packets = 2390
Received Packets = 2006
Lost Packets = 384
PDR = 83.9331%
Average Delay = 3.25752 ms
```

**Gambar 5.5 Pengujian Parameter pada Protokol DSDV**

Untuk mendapatkan nilai dari *routing overhead*, cara yang digunakan adalah dengan menggunakan bantuan *script* awk. Simulasi yang dijalankan akan menghasilkan file *trace* yang berisikan riwayat pengiriman paket selama simulasi. Pada Gambar 5.6 di bawah, *script* awk digunakan untuk membaca file *trace* tersebut. Setelah perintah berhasil dieksekusi, hasil dari penghitungan *routing overhead* akan muncul seperti yang nampak pada gambar di bawah.

```
root@prayudhi-X450JN:/home/prayudhi/Desltop/ns-3-allinone/ns-
3-dev# awk -f dsdv.awk DSDV_Uji_20Nodes.tr
Routing Overhead DSDV = 98786
root@prayudhi-X450JN:/home/prayudhi/Desltop/ns-3-allinone/ns-
3-dev# awk -f olsr.awk OLSR_Uji_20Nodes.tr
Routing Overhead OLSR = 51990
```

**Gambar 5.6 Pengujian Penghitungan *Routing Overhead***

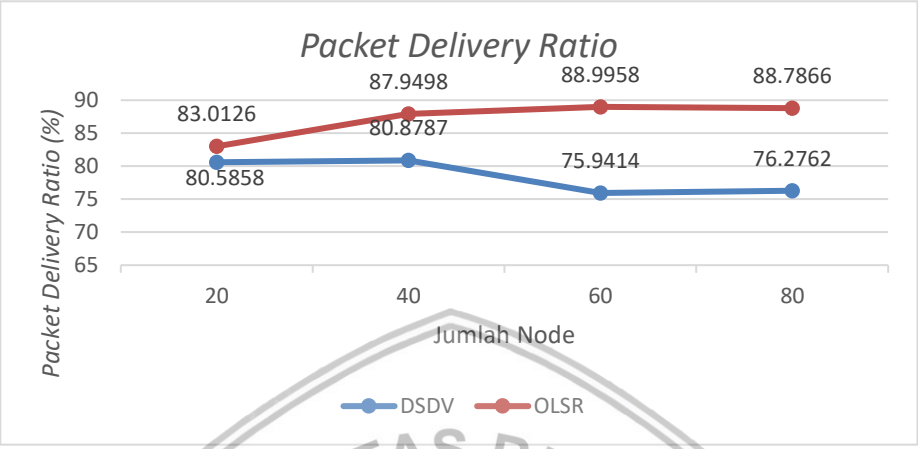
## 5.2 Analisis

### 5.2.1 Analisis Jumlah *Node* Terhadap Kinerja Protokol

Skenario pertama adalah simulasi dengan mengganti jumlah *node* yang digunakan tanpa mengganti nilai dari parameter lain seperti kecepatan minimum, kecepatan maksimum, luas area simulasi, waktu simulasi, dan ukuran paket. Simulasi dijalankan dengan jumlah *node* sebanyak 20 *node*, 40 *node*, 60 *node*, dan 80 *node* secara bergantian. Skenario ini bertujuan untuk melihat pengaruh jumlah *node* terhadap kinerja protokol DSDV dan OLSR. Terdapat tiga parameter yang digunakan untuk mengukur kinerja dari tiap protokol, yakni *packet delivery ratio*, *end-to-end delay*, dan *routing overhead*.

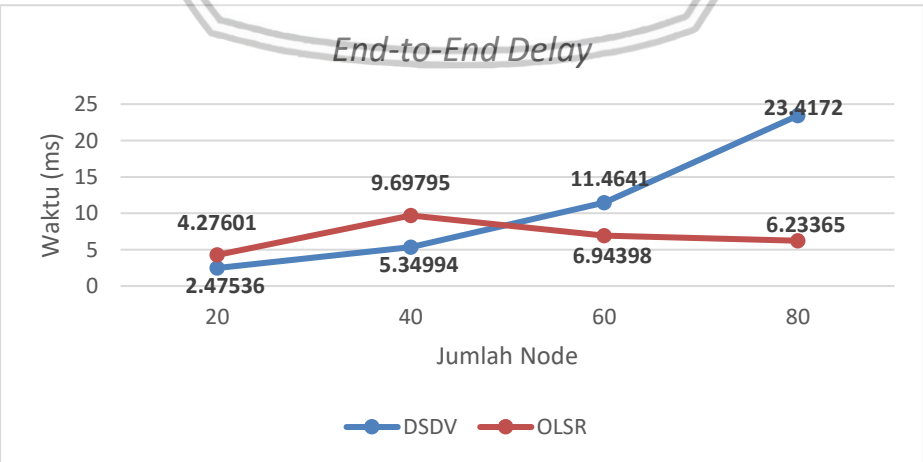
Pada Gambar 5.7 di bawah dapat dilihat bahwa protokol OLSR memiliki nilai *packet delivery ratio* yang lebih tinggi dibandingkan dengan protokol DSDV. Seiring dengan bertambahnya jumlah *node*, nilai *packet delivery ratio* pada protokol DSDV menurun. Pada simulasi dengan jumlah *node* 20, *packet delivery ratio* protokol DSDV menunjukkan nilai 80.5858%, sedangkan protokol OLSR menunjukkan nilai 83.0126%. Pada simulasi dengan jumlah *node* 40, nilai *pacekt delivery ratio* pada protokol DSDV sebesar 80.8787% dan pada protokol OLSR sebesar 87.9498%. Semakin bertambahnya jumlah *node* perbedaan kedua protokol semakin terlihat, karena *packet delivery ratio* pada protokol DSDV dengan jumlah *node* 60 mengalami penurunan yang cukup signifikan hingga mencapai 75.9414%.

Sedangkan protokol OLSR dengan jumlah *node* 60 memiliki nilai *packet delivery ratio* 88.9958%. Selisih nilai *packet delivery ratio* kedua protokol dengan jumlah *node* 80 masih besar, dengan nilai *packet delivery ratio* sebesar 76.2762% pada protokol DSDV dan sebesar 88.7866% pada protokol OLSR.



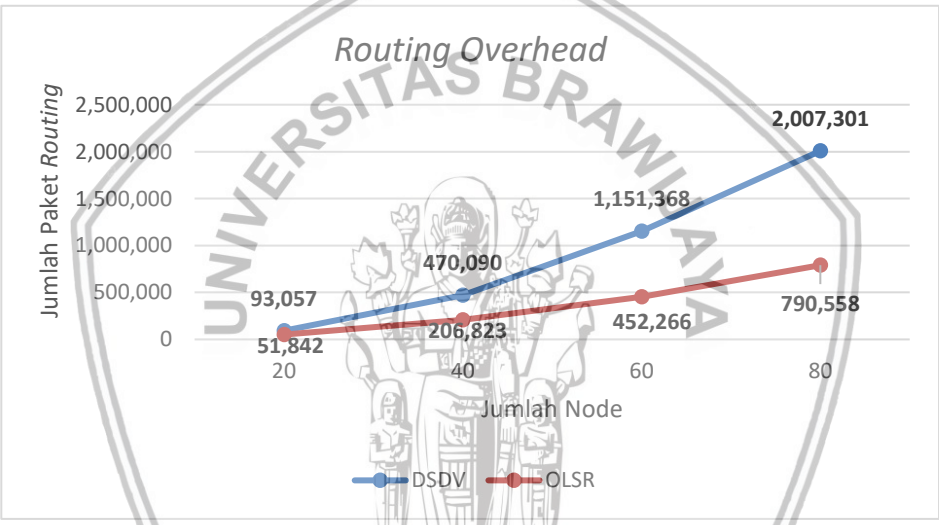

**Gambar 5.7 Pengaruh Jumlah *Node* Terhadap *Packet Delivery Ratio***

Pada Gambar 5.8 di bawah dapat dilihat bahwa perubahan jumlah *node* memiliki dampak yang berbeda terhadap nilai *end-to-end delay* pada kedua protokol. Pada simulasi dengan jumlah *node* 20, protokol DSDV memiliki nilai *end-to-end delay* sebesar 2.47536 ms sedangkan protokol DSDV sebesar 4.27601 ms. Begitu pula pada simulasi dengan jumlah *node* 40, nilai *end-to-end delay* pada protokol DSDV masih lebih rendah dibandingkan dengan protokol OLSR walaupun nilai *end-to-end delay* kedua protokol mengalami kenaikan. Namun pada simulasi dengan jumlah *node* 60, protokol DSDV mengalami kenaikan nilai *end-to-end delay* hingga 11.4641 ms sedangkan protokol OLSR mengalami penurunan nilai *end-to-end* delay hingga 6.94398 ms. Pada saat jumlah *node* 80, protokol OLSR tidak begitu banyak mengalami perubahan pada nilai *end-to-end delay*, sedangkan protokol DSDV mengalami peningkatan yang tajam hingga nilai *end-to-end delay* mencapai 23.4172 ms.




**Gambar 5.8 Pengaruh Jumlah *Node* Terhadap *End-to-End Delay***

Gambar 5.9 di bawah menunjukkan pengaruh jumlah *node* terhadap *routing overhead* pada kedua protokol. Pada umumnya perubahan jumlah *node* memiliki dampak yang sama pada kedua protokol, keduanya menunjukkan kenaikan jumlah paket *routing* yang dikirimkan seiring dengan bertambahnya *node*. Pada simulasi dengan jumlah *node* 20, *routing overhead* pada protokol DSDV menunjukkan nilai 93,057 sedangkan protokol OLSR menunjukkan nilai yang lebih rendah yakni 51,842. Saat simulasi dengan *node* 40, nilai *routing overhead* pada protokol DSDV meningkat hingga sekitar lima kali lipat yakni sebanyak 470,090 dan protokol OLSR meningkat hingaa 206,823. Nilai *routing overhead* kedua protokol terus meningkat, hingga saat simulasi dengan *node* sebanyak 80, nilai *routing overhead* pada protokol DSDV mencapai 2,007,301 dan protokol OLSR mencapai 790,558. Perbedaan *routing overhead* kedua protokol memang terlihat jelas pada setiap simulasi dengan kondisi *node* berjumlah 20 hingga kondisi terakhir dengan *node* berjumlah 80.




**Gambar 5.9 Pengaruh Jumlah *Node* Terhadap *Routing Overhead***

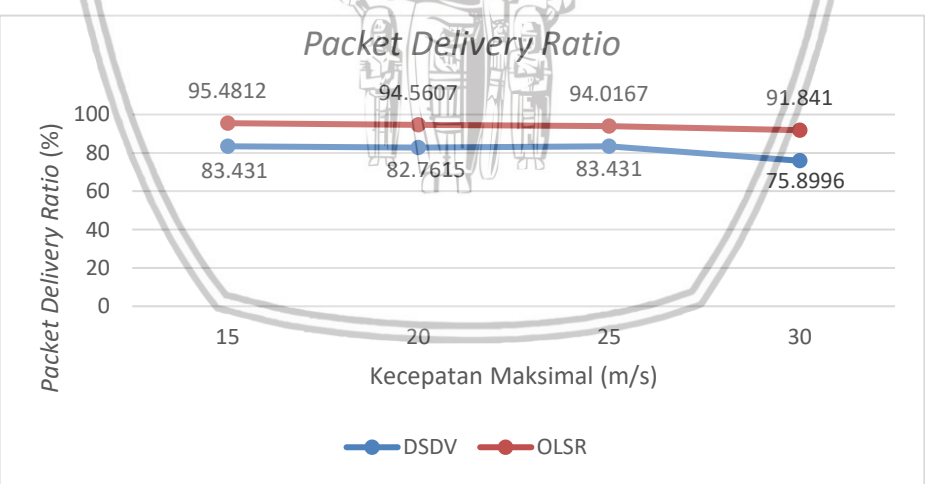
Berdasarkan hasil pengujian dengan skenario variasi jumlah *node*, dapat dilihat bahwa kedua protokol menunjukkan kinerja yang berbeda. Dengan bertambahnya jumlah *node* pada jaringan, protokol OLSR menunjukkan peningkatan nilai *packet delivery ratio* sedangkan pada protokol DSDV nilainya menurun seiring bertambahnya jumlah *node*. Hal tersebut berkaitan dengan nilai *routing overhead* kedua protokol. Pada protokol DSDV, nilai *routing overhead* selalu meningkat tajam tiap bertambahnya jumlah *node* sehingga jaringan akan terbebani dengan banyaknya paket *routing* yang harus dikirimkan ke tiap *node* dan membuat nilai *packet delivery ratio* menurun karena jaringan akan terbebani oleh paket *routing*. Tingginya nilai dari *routing overhead* juga mempengaruhi nilai *end-to-end delay* pada protokol DSDV. Meskipun jalur yang tersedia semakin bertambah karena menigkatnya jumlah *node* pada jaringan, nilai *routing overhead* yang sangat tinggi pada protokol DSDV menjadikan lalu lintas jaringan dipenuhi oleh paket *routing*. Oleh karena itu protokol OLSR lebih unggul pada parameter *end-to-end delay* saat variasi jumlah *node*. Berbeda dengan protokol OLSR yang memiliki nilai *routing overhead* yang berada jauh di bawah protokol DSDV karena protokol OLSR menggunakan MPR sehingga mengurangi *flooding* pada jaringan. Hal tersebut

yang membuat nilai *packet delivery ratio* pada protokol OLSR meningkat seiring bertambahnya jumlah *node* karena jalur yang dapat ditempuh untuk sampai ke tujuan semakin bertambah dan beban pada jaringan tidak sebesar protokol DSDV.

### 5.2.2 Analisis Kecepatan *Node* Terhadap Kinerja Protokol

Skenario kedua adalah simulasi dengan mengganti kecepatan maksimum yang digunakan tanpa mengganti nilai dari parameter lain seperti jumlah *node*, kecepatan minimum, luas area simulasi, waktu simulasi, dan ukuran paket. Simulasi dijalankan dengan kecepatan maksimum mencapai 15, 20, 25, dan 30 m/s secara bergantian. Skenario ini bertujuan untuk melihat pengaruh kecepatan *node* terhadap kinerja protokol DSDV dan OLSR. Terdapat tiga parameter yang digunakan untuk mengukur kinerja tiap protokol, yakni *packet delivery ratio*, *end-to-end delay*, dan *routing overhead*.

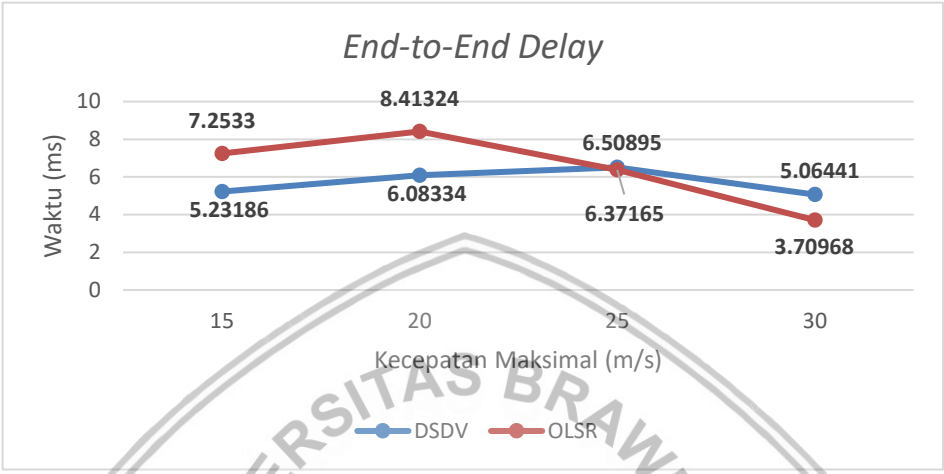
Pada Gambar 5.10 di bawah dapat dilihat bahwa seiring meningkatnya kecepatan *node* mengakibatkan nilai *packet delivery ratio* pada kedua protokol sedikit menurun. Saat simulasi dengan kecepatan maksimum sebesar 15 m/s, *packet delivery ratio* pada protokol DSDV menunjukkan nilai 83.431% sedangkan pada protokol OLSR menunjukkan nilai 95.4812%. Pada saat simulasi dengan kecepatan maksimum sebesar 20 m/s, nilai *packet delivery ratio* sedikit mengalami penurunan pada kedua protokol yakni menjadi 82.7615% pada protokol DSDV dan 94.5607% pada protokol OLSR. Pada saat simulasi dengan kecepatan maksimum sebesar 30 m/s, nilai *packet delivery ratio* pada protokol DSDV menjadi sebesar 75.8996% dan pada protokol OLSR menjadi sebesar 91.841%.




**Gambar 5.10 Pengaruh Kecepatan Maksimum Terhadap *Packet Delivery Ratio***

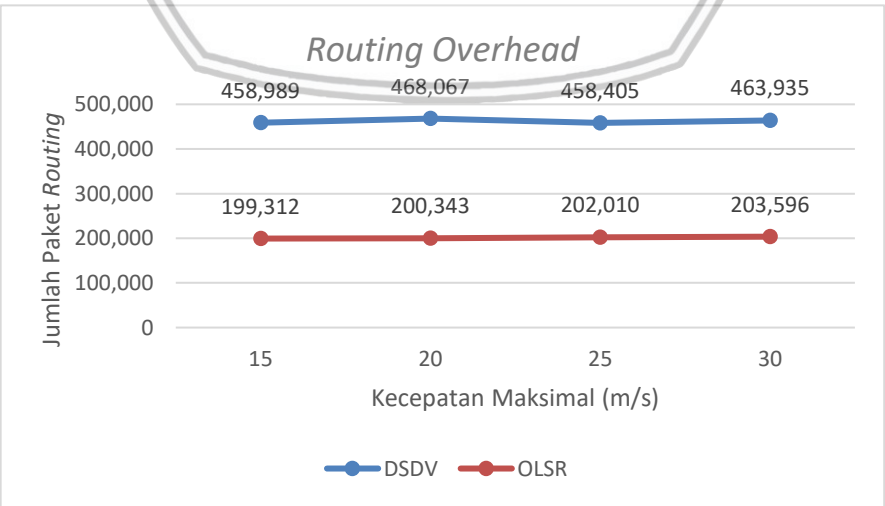
Pada Gambar 5.11 di bawah dapat dilihat bahwa perubahan kecepatan maksimum pada *node* menyebabkan terjadinya fluktuasi nilai *end-to-end delay* pada kedua protokol. Pada saat simulasi dengan kecepatan maksimum sebesar 15 m/s, rata-rata waktu *end-to-end delay* pada protokol DSDV adalah 5.23186 ms sedangkan rata-rata waktu *end-to-end delay* pada protokol OLSR adalah 7.2533 ms. Rata-rata waktu *end-to-end delay* pada kedua protokol meningkat saat pengujian dengan kecepatan maksimum sebesar 20 m/s, pada protokol DSDV menjadi sebesar 6.08334 ms dan pada protokol OLSR menjadi 8.41324 ms. Pada

simulasi dengan kecepatan maksimum sebesar 25 m/s, rata-rata waktu *end-to-end delay* pada protokol DSDV naik menjadi 6.50895 ms sedangkan pada protokol OLSR mengalami penurunan menjadi 6.37165 ms. Pada simulasi dengan kondisi terakhir dimana kecepatan maksimum sebesar 30 m/s, kedua protokol mengalami penurunan rata-rata waktu *end-to-end delay*, pada protokol DSDV nilainya menjadi 5.06441 ms dan pada protokol OLSR nilainya menjadi 3.70968 ms.




**Gambar 5.11 Pengaruh Kecepatan Maksimum Terhadap *End-to-End Delay***

Pada Gambar 5.12 di bawah dapat dilihat bahwa nilai routing overhead kedua protokol tidak banyak berubah. Pada saat simulasi dengan jumlah *node* sebanyak 15, nilai routing overhead pada protokol DSDV mencapai 458,989 sedangkan pada protokol OLSR hanya 199,312. Saat simulasi dengan jumlah *node* 20, protokol DSDV menunjukkan nilai routing overhead sebesar 468,067 sedangkan pada protokol OLSR sebesar 200,343. Pada kondisi jumlah *node* sebanyak 25 dan 30 nilai routing overhead kedua protokol tidak mengalami perubahan yang signifikan yakni sebesar 458,405 dan 463,935 pada protokol DSDV dan sebesar 202,010 dan 203,596 pada protokol OLSR.




**Gambar 5.12 Pengaruh Kecepatan Maksimum Terhadap *Routing Overhead***

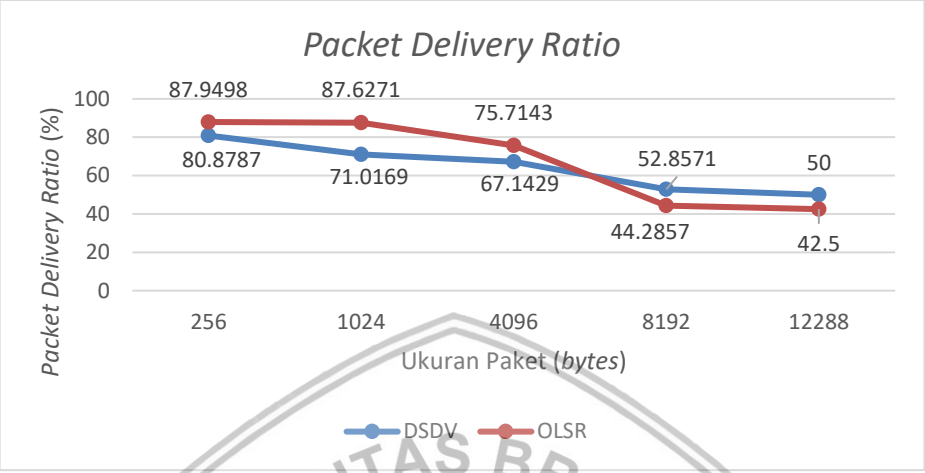
Secara umum kedua protokol mengalami penurunan nilai *packet delivery ratio* seiring dengan meningkatnya kecepatan maksimum pada *node*. Karena pada dasarnya semakin tinggi pergerakan *node* maka rasio pengiriman paketnya pun akan menurun. Jika dilihat dari grafik yang telah ditunjukkan sebelumnya, protokol OLSR lebih unggul dalam hal *packet delivery ratio*. Berbeda dengan nilai *end-to-end delay* yang mengalami fluktuasi pada skenario kedua ini. Dalam penelitian ini, pada setiap pengujiannya digunakan 10 *node* yang bertugas sebagai pengirim dan 10 *node* yang bertugas sebagai penerima sehingga terdapat 10 *flow* pengiriman. Fluktuasi *end-to-end delay* pada penelitian ini disebabkan karena pergerakan *node* yang sifatnya acak sehingga menyebabkan nilai *end-to-end delay* dari tiap *flow* bervariasi. Misalnya saja *node* pengirim dan penerima pada beberapa *flow* posisinya selalu berdekatan saat simulasi pertama, dan cenderung berjauhan pada simulasi selanjutnya. Untuk nilai dari *routing overhead* pada skenario kedua ini sebenarnya tidak banyak mengalami perubahan jika dibandingkan dengan skenario pertama yakni variasi jumlah *node*. Secara garis besar, pada skenario kedua ini nilai routing overhead semakin bertambah seiring bertambahnya kecepatan *node*. Namun pada beberapa kasus, nilai *routing overhead* mengalami sempat mengalami penurunan. Hal ini disebabkan karena penelitian ini menggunakan protokol 802.11b dimana luas area cakupannya mencapai 45 meter jika di dalam ruangan dan dapat mencapai ratusan meter jika di luar ruangan. Cakupan protokol 802.11b dapat dikatakan cukup besar dalam area simulasi yang besarnya hanya 400x400 meter. Itulah mengapa jumlah paket *routing* tidak begitu banyak mengalami perubahan karena sebagian besar *node* mungkin masih dalam area cakupannya. Walaupun demikian, nilai *routing overhead* protokol OLSR lebih unggul karena menggunakan MPR untuk mengurangi beban *overhead* pada jaringan.

### 5.2.3 Analisis Ukuran Paket Terhadap Kinerja Protokol

Skenario ketiga adalah simulasi dengan mengganti ukuran paket yang dikirim tanpa mengganti nilai dari parameter yang lain seperti jumlah *node*, kecepatan minimum, kecepatan maksimum, luas area simulasi, dan waktu simulasi. Simulasi dijalankan dengan ukuran paket sebesar 256, 1024, 4096, 8192, dan 12288 bytes. Skenario ini bertujuan untuk melihat pengaruh ukuran paket yang dikirimkan terhadap kinerja protokol DSDV dan OLSR.

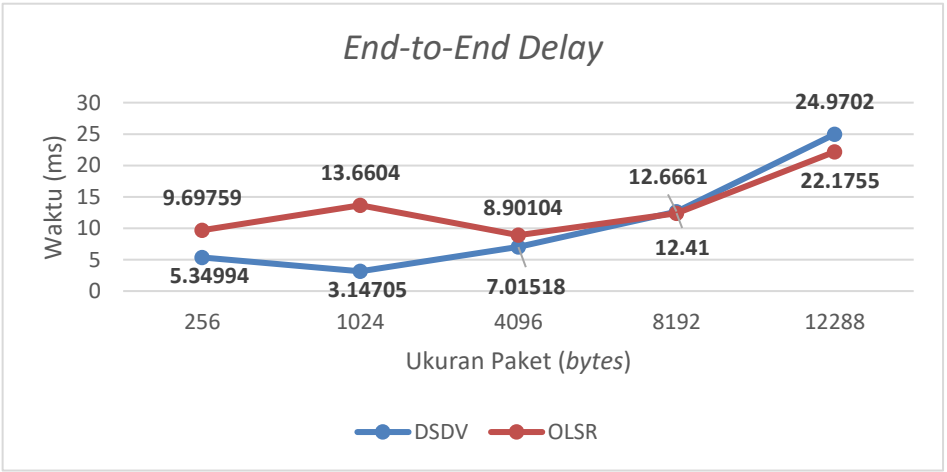
Pada Gambar 5.13 dapat dilihat pengaruh ukuran paket yang dikirimkan dengan nilai *packet delivery ratio*. Semakin besar ukuran paket maka nilai *packet delivery ratio* semakin menurun. Pada simulasi dengan ukuran paket sebesar 256 ukuran paket sebesar 256 *bytes*, nilai *packet delivery ratio* pada protokol DSDV adalah 80.8787% sedangkan pada protokol OLSR adalah 87.9498%. Pada saat simulasi dengan ukuran paket sebesar 1024 *bytes*, nilai *packet delivery ratio* pada kedua protokol mengalami penurunan yakni menjadi 71.0169% pada protokol DSDV dan 87.6271% pada protokol OLSR. Begitu pula pada simulasi dengan ukuran paket sebesar 4096 *bytes*, nilai *packet delivery ratio* kedua protokol mengalami penurunan menjadi 67.1429% pada protokol DSDV dan 75.7143% pada protokol OLSR. Pada saat simulasi dengan ukuran paket sebesar 8192 *bytes*, kedua protokol

mengalami penurunan nilai *packet delivery ratio* yang cukup jauh menjadi 52.8571% pada protokol DSDV dan 44.2857% pada protokol OLSR. Dan pada simulasi dengan ukuran paket sebesar 12288 *bytes*, nilai *packet delivery ratio* pada protokol DSDV menjadi 50% dan pada protokol OLSR menjadi 42.5%.



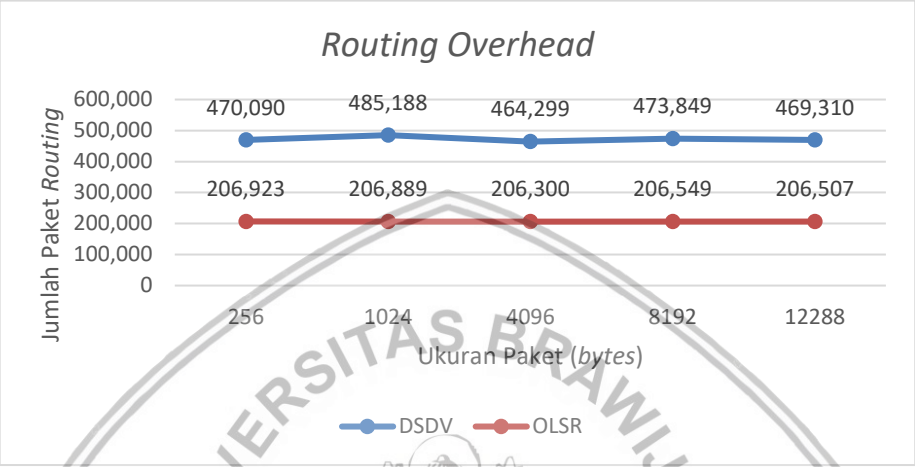

**Gambar 5.13 Pengaruh Ukuran Paket Terhadap *Packet Delivery Ratio***

Pada Gambar 5.14 di bawah dapat dilihat pengaruh ukuran paket terhadap nilai *end-to-end delay* pada kedua protokol. Secara umum nilai *end-to-end delay* pada kedua protokol mengalami peningkatan seiring dengan bertambahnya ukuran paket. Saat simulasi dengan ukuran paket sebesar 256 *bytes*, nilai *end-to-end delay* pada protokol DSDV sebesar 5.34994 ms dan pada protokol OLSR sebesar 9.69759 ms. Pada simulasi dengan ukuran paket sebesar 1024 *bytes,* nilai *end-to-end delay* pada protokol DSDV sebesar 3.14705 ms dan pada protokol OLSR sebesar 13.6604 ms. Pada simulasi dengan ukuran paket sebesar 4096 *bytes*, nilai *end-to-end delay* pada protokol DSDV sebesar 7.01518 ms dan pada protokol OLSR sebesar 8.90104 ms. Pada simulasi dengan ukuran paket sebesat 8192 *bytes*, nilai *end-to-end delay* pada protokol mengalami kenaikan hingga menjadi 12.6661 ms berada sedikit di atas protokol OLSR dengan nilai sebesar 12.41 ms. Dan pada simulasi dengan ukuran paket sebesar 12288 *bytes*, nilai *end-to-end delay* pada protokol DSDV sebesar 24.9702 ms dan pada protokol OLSR sebesar 22.1755 ms.




**Gambar 5.14 Pengaruh Ukuran Paket Terhadap *End-to-End Delay***

Pada Gambar 5.15 di bawah menunjukkan pengaruh ukuran paket yang dikirim dengan nilai *routing overhead* pada kedua protokol. Pada umumnya ukuran paket yang dikirimkan tidak begitu mempengaruhi nilai dari *routing overhead* pada protokol DSDV maupun pada protokol OLSR. Seiring bertambahnya ukuran paket, nilai *routing overhead* pada protokol DSDV berturut-turut adalah 470,090, 485,188, 464,299, 473,849, dan 469,310. Sedangkan pada protokol OLSR nilainya secara berturut-turut adalah 206,923, 206,889, 206,300, 206,549, dan 206,507.




**Gambar 5.15 Pengaruh Ukuran Paket Terhadap *Routing Overhead***

Pengujian pada skenario ketiga ini menunjukkan pengaruh ukuran paket yang dikirim terhadap kinerja kedua protokol. Secara garis besar nilai *packet delivery ratio* pada kedua protokol mengalami penurunan seiring dengan meningkatnya ukuran paket dan protokol OLSR lebih unggul dibandingkan protokol DSDV dengan rata-rata PDR 67,61%. Nilai PDR terendah dari kedua protokol terdapat pada simulasi dengan ukuran paket terbesar, yakni 12288 *bytes*. Menurunnya nilai PDR disebabkan jaringan yang terbebani karena harus mengirim paket-paket berukuran besar. Bertambahnya ukuran paket yang dikirimkan tentu akan mempengaruhi nilai *end-to-end delay*. Secara umum, nilai *end-to-end delay* pada kedua protokol mengalami kenaikan walaupun pada beberapa kasus terdapat penurunan nilai *end-to-end delay*. Penurunan dapat terjadi karena beberapa paket dengan *delay* yang sangat besar tidak dapat diterima oleh *node* tujuan sehingga nilai *delay* dari paket-paket tersebut tidak dihitung. Dalam hal nilai *end-to-end delay*, protokol DSDV lebih unggul daripada protokol OLSR. Pada skenario dengan variasi ukuran paket tidak memberikan dampak yang besar terhadap nilai *routing overhead*. Pada skenario ini, nilai *routing overhead* mengalami naik turun namun perubahannya tidak terlalu besar. Hal ini terjadi karena berapapun ukuran paket yang dikirimkan tidak mempengaruhi seberapa sering paket *routing* dikirimkan. Selain itu juga, seperti yang telah dijelaskan pada bagian sebelumnya bahwa pada penelitian ini menggunakan protokol 802.11b dengan area jangkauan yang cukup luas. Itulah mengapa nilai *routing overhead* pada skenario ini tidak banyak mengalami perubahan. Sama seperti pada dua skenario yang sebelumnya, protokol OLSR lebih unggul dibandingkan protokol DSDV dalam hal *routing overhead*.

# BAB 6 PENUTUP

## 6.1 Kesimpulan

Setelah didapatkan hasil pengujian dan analisis dari penelitian yang telah dilakukan, kesimpulan yang dapat diambil adalah:

1. Implementasi protokol DSDV dan protokol OLSR dengan pola pergerakan *gauss-markov* dalam jaringan MANET pada *Network Simulator 3* dapat bekerja sesuai dengan skenario yang ditentukan. Skenario yang dimaksud adalah skenario dengan variasi jumlah *node*, skenario dengan variasi kecepatan maksimal *node*, dan skenario dengan variasi ukuran paket.
2. Terdapat tiga skenario pada penelitian ini, yakni skenario dengan variasi jumlah *node*, skenario dengan variasi kecepatan maksimal, dan skenario dengan variasi ukuran paket. Berikut merupakan kinerja protokol dari tiap skenario yang diujikan:
    a. Pada skenario pertama yakni variasi jumlah *node*, kedua protokol menunjukkan adanya dampak yang besar pada nilai *packet delivery ratio, end-to-end delay,* dan *routing overhead*. Seiring dengan bertambahnya jumlah *node*, protokol DSDV secara garis besar menunjukkan penurunan kinerja. Berbeda dengan protokol OLSR yang justru menunjukkan peningkatan kinerja. Meskipun protokol OLSR juga mengalami kenaikan nilai *routing overhead,* namun nilainya masih tetap lebih baik dibandingkan dengan protokol DSDV.
    b. Pada skenario kedua terlihat bahwa kecepatan maksimal *node* tidak memberikan pengaruh besar terhadap nilai *packet delivery ratio* dan *routing overhead* pada kedua protokol dan jika dilihat dari dua parameter uji tersebut, protokol OLSR menunjukkan kinerja yang lebih baik pada semua variasi kecepatan maksimal *node*. Kecepatan maksimal *node* juga berdampak pada nilai *end-to-end delay*. Saat kecepatan maksimal *node* sebesar 20 m/s dan 40 m/s, nilai *end-to-end delay* pada protokol DSDV lebih rendah dibandingkan protokol OLSR. Namun protokol OLSR lebih unggul saat kecepatan maksimal *node* sebesar 60 m/s dan 80 m/s.
    c. Pada skenario ketiga dengan variasi ukuran paket yang dikirimkan, kedua protokol mengalami penurunan nilai *packet delivery ratio*. Penurunan nilai yang sangat besar terjadi saat ukuran paket sebesar 8192 *bytes,* pada protokol DSDV nilai *packet delivery ratio* turun menjadi 52.85% dan pada protokol OLSR nilainya menjadi 44.28% di bawah protokol DSDV. Protokol DSDV unggul pada nilai *end-to-end delay* saat simulasi dengan ukuran paket sebesar 256, 1024, dan 4096 *bytes*. Namun pada saat ukuran paket sebesar 8192 dan 12288 *bytes* nilai *end-to-end delay* protokol OLSR sedikit lebih baik dibandingkan

protokol DSDV. Nilai *routing overhead* kedua protokol tidak banyak mengalami perubahan.

3. Dengan mengacu pada nilai *packet delivery ratio, end-to-end delay,* dan *routing overhead*, OLSR memiliki kinerja yang lebih baik dibandingkan dengan protokol DSDV pada skenario variasi jumlah *node*. Semakin bertambahnya jumlah *node*, perbedaan kinerja kedua protokol semakin jauh. Pada umumnya protokol OLSR memiliki performa yang lebih baik daripada protokol DSDV pada saat pengujian dengan jumlah *node* dan kecepatan maksimal menengah ke atas. Sedangkan protokol DSDV memiliki keunggulan nilai *end-to-end delay* pada saat pengujian dengan jumlah *node* dan kecepatan maksimal menengah ke bawah. Kedua protokol memiliki kelebihan masing-masing tergantung pada karakterisktik jaringan yang digunakan pada saat simulasi.

## 6.2 Saran

Setelah melakukan penelitian ini, beberapa saran yang dapat dijadikan bahan pertimbangan untuk pengembangan penelitian yang akan datang adalah:

1. Penelitian berikutnya dapat dikembangkan dengan menggunakan protokol yang berbeda serta parameter yang menjadi tolak ukur kinerja protokol dapat lebih beragam.
2. Perlu dilakukan uji coba kinerja protokol pada *simulator* yang berbeda.

# DAFTAR PUSTAKA


Aho, A., Kernighan, B. & Weinberger, P. 1979. Awk - a pattern scanning and processing language. *Software: Practice and Experience*, pp.267-279.

Andria Siregar, R. 2016. Peningkatan Kinerja Protokol Optimized Link State Routing Pada Lingkungan Vehicular Adhoc Network dengan Menggunakan Prediksi Mobilitas dan Multipath Routing. Tesis. Program Magister Institut Teknologi Sepuluh Nopember Surabaya.

Camp, T., Boleng, J. & Davies, V. 2002. A survey of mobility models for ad hoc network research. 2002 *Wireless Communications and Mobile Computing*, pp. 483-502.

Clausen, T. & Jacquet, P. 2003. Optimized link state routing protocol (OLSRP).

Daas, A., Mofleh, K., Jabr, E. dan Hamad, S. 2015. Comparison between AODV and DSDV Routing protocols in Mobile Ad-hoc Network (MANET). *2015 5th National Symposium on Information Technology*.

Fatkhurrozi, F., Widasari, E. & Bhawiyuga, A. 2018. Analisis Perbandingan Kinerja Protokol AOMDV, DSDV, Dan ZRP Sebagai Protokol Routing Pada Mobile Ad-Hoc Network (MANET). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, pp.3671-3680.

Gowrishankar, S. & Basavaraju, T. 2010. Analysis of AOMDV and OLSR Routing Protocols Under Levy-Walk Mobility Model and Gauss-Markov Mobility Model for Ad Hoc Networks. *International Journal on Computer Science and Engineering*, pp.979-986.

Gupta, P. 2016. A Literature Survey of MANET. 2016 *International Research Journal of Engineering and Technology*, pp.95-99.

Hazra, S. & Setua, S. 2010. Optimization on Control Overhead in MANET. *International Journal of Computer Applications*, 12(4), pp.16-21.

Hogie, L., Bouvry, P. & Guinand, F. 2006. An Overview of MANETs Simulation. *Electronic Notes in Theoretical Computer Science*, 150(1), pp.81-101.

Manpreet, & Malhotra, J., 2014. A Survey on MANET Simulation Tools. 2014 *International Conference on Innovative Applications of Computational Intelligence on Power, Energy and Controls with their Impact on Humanity*, pp.496-498.

Narra, H., Cheng, Y., Centikaya, E., Rohrer, J. & Sterbenz, J. 2011. Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3. *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pp.439-446.

R, S. & Kumar, N. 2014. A Survey on Proactive Routing Protocols in MANETs. *International Conference on Science, Engineering and Management Research (ICSEMR)*, pp.1-7.

Raja, L. & Baboo, S. 2014. An Overview of MANET: Applications, Attacks and Challenges. *International Journal of Computer Science and Mobile Computing*, pp.408-417.

Rohal, P., Dahiya, R., & Dahiya, P. 2013. Study and Analysis of Throughput, Delay and Packet Delivery Ratio in MANET for Topology Based Routing Protocols (AODV, DSR and DSDV). *International Journal for Advance Research in Engineering and Technology*, pp.54-58.

Saraswat, B., Bhardwaj, M. & Pathak, A. 2015. Optimum Experimental Results of AODV, DSDV & OLSR Routing Protocol in Grid Environment. *Procedia Computer Science*, pp.1359-1366.

Shaukat, K. & Syrotiuk, V. 2013. Using local conditions to reduce control overhead. Ad Hoc Networks, 11(6), pp.1782-1795.